National Defence / Défense nationale

A SIMULATION MODEL FOR DETECTING AIRBORNE TARGETS BY RADAR

by

S.Y. Slinn
Space-Based Radar Project Office
Electronics Division

M. Lovell and D. Miller
Co-op Students

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
TECHNICAL NOTE 86-13

PCN
33100

June 1986
Ottawa

ABSTRACT

In support of the Space Based Radar (SBR) surveillance system technologies, a satellite simulation model has been developed. This simulation models the orbital dynamics of multiple radar satellites, simulates the radar beam in order to study the surveillance coverage capabilities of the satellites and finally, simulates the detection of airborne targets. This report summarizes the major mathematical concepts involved in the model and provides an overview of the computer implementation of this model.

RÉSUMÉ

Un modèle de simulation de satellites a été développé dans le cadre des travaux de soutien de la technologie du système de surveillance fondé sur le radar en orbite spatiale (SBR). Cette simulation modélise les caractéristiques dynamiques orbitales des systèmes à plusieurs satellites radar, simule le faisceau radar de manière à permettre l'étude des caractéristiques de la couverture de surveillance des satellites, et, finalement, simule la détection de cibles en vol. Le rapport résume les principaux concepts mathématiques ayant servi à la réalisation du modèle et contient une vue d'ensemble de l'implantation informatique du modèle.

TABLE OF CONTENTS

	Page
1.0 INTRODUCTION	1
2.0 SATELLITE SIMULATION MODEL	1
2.1 Orbital Model	1
2.1.1 Orbital Parameters	2
2.2 Satellite Swaths	2
2.3 Target Detection	5
2.4 Graphical Output	10
3.0 IMPLEMENTATION OF MATHEMATICAL MODEL	14
3.1 Drawglobe	18
3.2 Satsat	19
3.2.1 General Routines	19
3.2.2 Computational Routines	19
3.2.3 Input/Output Routines	21
3.3 Swath	21
3.3.1 General Routines	23
3.3.2 Computational Routines	23
3.3.3 Input/Output Routines	24
3.4 Satpath	25
3.4.1 General Routines	25
3.4.2 Computational Routines	26
3.4.3 Input/Output Routines	26
3.5 Erase	27
4.0 SUMMARY DISCUSSION	29
REFERENCES	29
APPENDIX A: COORDINATE SYSTEMS AND COORDINATE TRANSFORMATIONS	31
A.1 Coordinate Systems	33
A.2 Coordinate Transformations	33 37

A.2.1 Orbital to Inertial Cartesian coordinates 38
A.2.2 Inertial Cartesian to Earth Rotating coordinates 40
A.2.3 Geocentric to Geodetic coordinates . . . 41
A.2.4 Cartesian to Spherical coordinates . . . 43
A.2.5 Vehicle Centred to Earth Centred Inertial Coordinates 45

A.3 Rotation of Coordinate Axes vs. Rotation of Vectors 47

APPENDIX B: METHOD OF TARGET POSITION DETERMINATION . . 51

APPENDIX C: USERS GUIDE 55

C.1 Compilation and Linking 55
C.2 Running the Programs 57

C.2.1 How to Run Drawglobe 57
C.2.2 How to Run Sgpsat 58

C.2.2.1 Epoch Parameters 58
C.2.2.2 Running the SGPSAT Program . . . 61

C.2.3 How to Run Swath 61
C.2.4 How to Run Satpath 63
C.2.5 How to Run Erase 64

C.3 Sample Run of the Simulation Package 64

APPENDIX D: DATA DICTIONARY OF ARRAYS AND FILES 69

D.1 Sgpdat Array 70
D.2 Fd Array 70
D.3 Filedata, Data Arrays and Geocor.dat File . . .
D.4 Trgcor.dat File 72
D.5 Detectpos.dat File 72

APPENDIX E: ESCAPE SEQUENCES 73

APPENDIX F: PROGRAM LISTINGS 75

LIST OF FIGURES

		Page
Figure 1:	Classical Element Set	4
Figure 2:	Radar Area of Illumination	6
Figure 3a:	Circle Defined on an Arbitrary Coordinate System	8
Figure 3b:	Top View	8
Figure 4a:	Vector Representations of Satellite and Target Positions	11
Figure 4b:	Vector Representations of Satellite and Target Positions with Respect to the Earth	11
Figure 5:	Target Position on a Spherical Earth	12
Figure 6:	Satellite and Target Positions on a Spherical Earth	12
Figure 7:	Polar View	15
Figure 8:	Mercator View	15
Figure 9:	Orbit Trace	16
Figure 10a:	Polar View of Ground Trace	16
Figure 10b:	Mercator View of Ground Trace	17
Figure 11:	Static Display of Target Detection	17
Figure 12:	SGPSAT Calling Sequence Hierarchy	20
Figure 13:	SWATH Calling Sequence Hierarchy	24
Figure 14:	SATPATH Calling Sequence Hierarchy	26
Figure A1:	Inertial Cartesian Reference Frame	34
Figure A2:	Earth Rotating Reference Frame	34
Figure A3:	Geographic Reference Frame	35
Figure A4:	Orbit Reference Frame	36
Figure A5:	Vehicle Centred Reference Frame	38
Figure A6:	Relationship between the Inertial Cartesian and Orbital Reference Frames	39
Figure A7:	Hour Angle	41
Figure A8:	Geocentric vs. Geodetic Latitude	42
Figure A9:	Earth Cartesian Coordinate System	43
Figure A10:	Vehicle Centred and Earth Centred Inertial Coordinate System	45
Figure A11:	Rotation of Axes about the Y-Axis	48
Figure A12:	Rotation of Vectors around the Y-Axis	49
Figure B1:	Constructing a Spherical Triangle	51
Figure B2:	Enlarged View of the Spherical Triangle	52
Figure B3:	Target Bearing	54
Figure C1:	Sample Output	67
Figure D1:	GEOCOR.DAT file	71
Figure D2:	TRGCOR.DAT file	72

1.0 INTRODUCTION

The Department of National Defence is undertaking initiatives to investigate the technologies associated with a Space Based Radar (SBR) surveillance system. A project office has been established at DREO to manage these initiatives. In support of this project office, a research group has been formed to develop and study SBR concepts at the systems level. A primary function of this group is to develop analytical tools and facilities to support these concepts and future systems trade off studies.

This report documents the first in a family of such analytical tools, a satellite simulation model. The mathematical basis of the simulation model has been described elsewhere [1,2,3] and is only outlined in the following discussion. The intention of this report is to summarize the major mathematical concepts involved in the model and provide an overview of the computer implementation of the model. Program operating instructions and software listings are given in the attached appendices.

2.0 SATELLITE SIMULATION MODEL

The purpose of the work documented in this report is to develop and exercise a model which would form the basis for the future development of mathematical models specific to SBR study requirements. Therefore, step one is to simulate the orbital path of a satellite or family of satellites in constellation to meet SBR mission requirements. The next step in the model development is to simulate the radar beam forming models in order to study the surveillance coverage capabilities of the satellite or constellation of satellites. Finally, the detection capability of the system thus defined must be described.

In the following discussion, as noted previously, the analytical model development associated with step one is described. A radar beam forming model has not yet been developed. Instead, and to allow for this future development, a simple ground swath model, based on geometry consideration only, has been incorporated into the satellite

simulator. Similarily, the model does not yet contain a detection model. A simple target detection model based upon the target being within the viewing cone of the satellite provides a convenient "hook" for future model refinement.

The overview of the satellite simulation model concepts and analytics are now described in terms of the orbital model, the coverage model and the detection model. Since the overall approach in the analytical development is one of user interaction with the facilities, extensive emphasis has been placed upon computer graphics as a study tool. The graphic output features of the model are described.

2.1 ORBITAL MODEL

The purpose of the orbital model is to simulate the orbital path of a satellite around the earth. A simple Keplerian model could have been chosen to predict a satellite's position in the orbit, but it would only be accurate if the gravitational attraction from a spherical earth is the only force acting upon a satellite. However, satellites are affected by atmospheric drag, gravitational attraction due to a non-spherical earth, the moon, and other perturbations. The orbital model that has been utilized is based on a paper written by Kozai [3]. This model assumes that air-drag is absent and that the earth's gravitational field is symmetric with respect to the axes. The orbital model deals with general perturbation theories which are analytical methods based on an analytic solution of the equations of motion. It is a second order theory using mean orbital elements that deal with periodic perturbations of the first order and secular perturbations up to the second order. A full description of the theory of this orbital model is given in a separate report. An overview of the model's concepts and analytics are presented here.

Throughout the development of this model, various coordinate systems have been used; the Inertial Cartesian, Earth Rotating, Geographic, Orbital and the Vehicle Centred coordinate system. At various times in the development of the model, it is necessary to change from one coordinate system to another. This is necessary because some calculations are performed more 'naturally' in one coordinate system but then have to be transformed in order to ensure that subsequent calculations all be performed referencing the same system. Appendix A contains a complete

description of all the coordinate systems used in the model along with a derivation of all the transformations utilized.

2.1.1 Orbital Parameters

There is more than one set of orbital elements that can be used to describe a satellite orbit, for example: the Inertial set and the Keplerian (or Classical) set.

Before introducing the orbital parameters, a few basic terms should be defined.

apogee - the point on the orbital plane that is furthest away from the earth

perigee - the point on the orbital plane that is closest to the earth

node - point of intersection between the orbital plane and the equatorial plane. There are two nodes, the ascending node (satellite goes from south to north) and the descending node (satellite goes from north to south).

epoch - the point in time in which the classical element parameters are defined.

The elements in the Inertial set are defined by the satellite position and velocity vectors, (Rx, Ry, Rz) and (Vx, Vy, Vz) respectively. However, these six variables are not unique and they are continually varying for a satellite as it progresses through its orbit.

The Keplerian or Classical set of orbital elements is used in this model. This set is comprised of five constant parameters called orbital elements which describe the shape, size and orientation of the orbit along with a sixth varying element, which is used to locate the position of the satellite in the orbit. The classical set of orbital elements are illustrated in Figure 1. They are as follows:

- a, semi-major axis - constant defining the size of the orbit
- e, eccentricity - constant defining the shape of the orbit
- i, inclination - angle between the orbit plane and

the earth's equatorial plane.

- Ω , longitude of ascending node - angle , measured counterclockwise, between the Vernal Equinox direction and the ascending node of the orbit.
- ω , argument of perigee - the angle, in the orbit plane, between the ascending node and the perigee point.
- γ_o , true anomaly - the angle, in the orbit plane, between the perigee point and the position of the satellite at epoch.

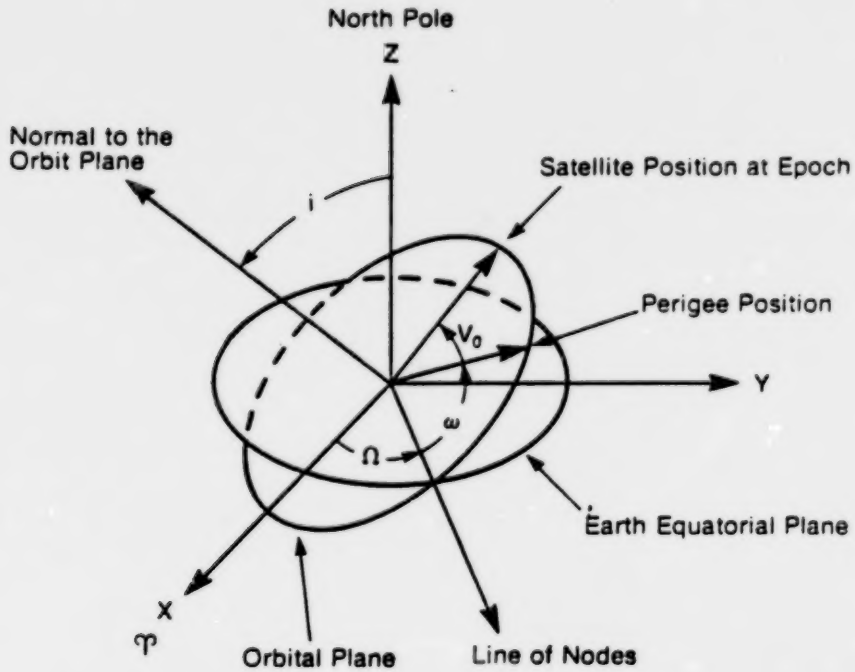


Figure 1: Classical Element Set

The first five classical elements were stated to be constant. In reality, they vary ove. a long period of time due to orbital perturbations. A perturbation is a deviation from some normal or expected motion. Perturbations are caused by other mass bodies such as the moon and additional forces such as a nonspherical earth.

There are three types of perturbations in the model, periodic, secular and atmospheric drag. Periodic perturbations follow a cycle and in turn are comprised of two components; short and long period perturbations. Secular perturbations increase or decrease continually over time. These two types of perturbations are conservative. However, this model also considers the non-conservative effect that atmospheric friction has on satellites. Each of the six orbital elements described above is affected by one or both of these types of perturbations.

A satellite is disturbed from its Keplerian orbit by the disturbing function, R.

$$R = U - \mu/r$$

This disturbing function can be expressed in terms of the orbital elements and separated into its component elements for the short and long periodic perturbations and the secular perturbations. The perturbation equations for each of the six orbital elements are generated. These are the equations that are used to update the orbital elements. A detailed description of the development of the perturbation equations is given at Reference 8.

2.2 SATELLITE SWATHS

This report does not involve radar concepts as a radar beam model will be the subject of future work. A simple ground swath model, based on geometry considerations only, is currently used in the satellite simulation model.

Therefore, according to this simplistic model, a target is detected by a satellite's radar when the target is within the radar's area of illumination. The boundaries of this area are determined by two angles, the minimum and maximum grazing angles. These angles specify the angle made from the radar to the surface of the earth, refer to Figure 2. As these angles are varied, the area of illumination gets larger or smaller. Detection occurs when the target is within the area bounded by the minimum and maximum grazing angles. This is the shaded area shown in Figure 2. The size of the grazing angle has a direct effect on the half angle of illumination at the centre of the earth shown in Figure 2. This angle is required to calculate the inner and outer circle defining the area of illumination. The meaning

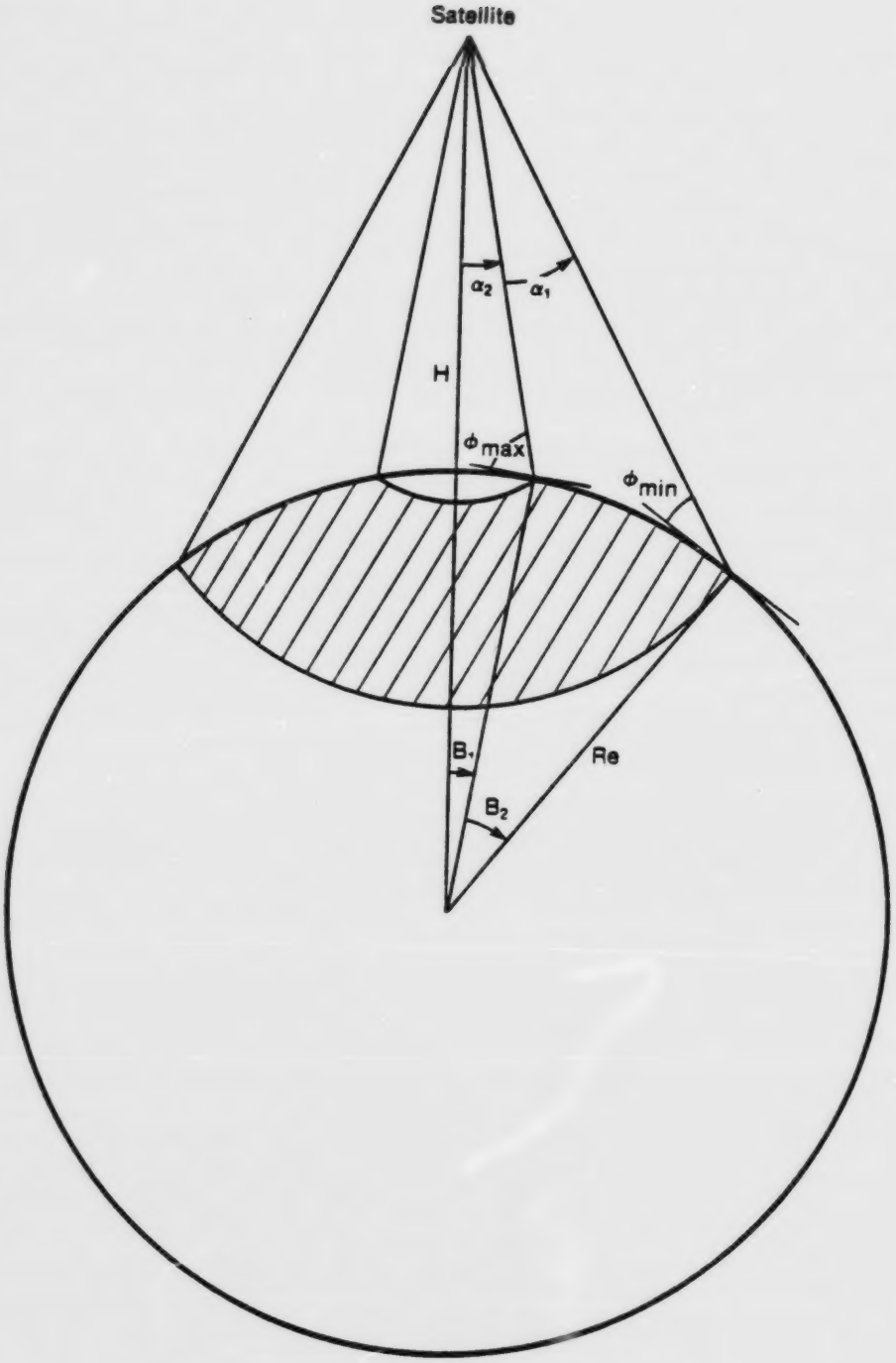


Figure 2: Radar Area of Illumination

of the symbols used in Figure 2 are as follows:

ϕ min - minimum grazing angle
ϕ max - maximum grazing angle
β_1 - half angle of illumination at centre of the earth for the minimum grazing angle
β_2 - half angle of illumination at centre of the earth for the maximum grazing angle
α_1 - half angle of illumination at satellite for ϕ min
α_2 - half angle of illumination at satellite for ϕ max
Re - radius of the earth
H - height of the satellite above the earth's surface

The derivation of the half angle of illumination is as follows.

From Figure 2, $\beta + \phi + \alpha + 90 = 180$

$\beta + \phi + \alpha = 90$

Let $\beta + \phi = \theta$

Using the Sine law,

$$(Re + H) / \sin(\phi + 90) = Re / \sin(\alpha)$$

$$(Re + H) / \cos(\phi) = Re / \sin(\alpha)$$
$$= Re / \sin(90 - \theta)$$
$$= Re / \cos(\theta)$$
$$= Re / \cos(\beta + \phi)$$

$$Re \times \cos(\phi) = (Re + H) \times \cos(\beta + \phi)$$

$$(Re \times \cos(\phi)) / (Re + H) = \cos(\beta + \phi)$$

$$(\cos^{-1}(Re \times \cos(\phi))) / (Re + H) = \beta + \phi$$

Therefore, $\beta = \cos^{-1}((Re \times \cos(\phi)) / (Re + H)) - \phi$

Initially, assume that the angle of revolution for the radar is 360 degrees, angle ϕ, in Figure 3a. This simplifies the definition of the area of illumination boundaries as they will now be full circles. First, these two boundary circles are defined as illustrated in Figure 3a. A top view of this circle is illustrated in Figure 3b. The X, Y, and Z coordinates of each point on this circle are defined as follows:

From Figure 3a,

$$z = Re \times \cos(\theta)$$

Let: $R = Re \times \sin(\theta)$

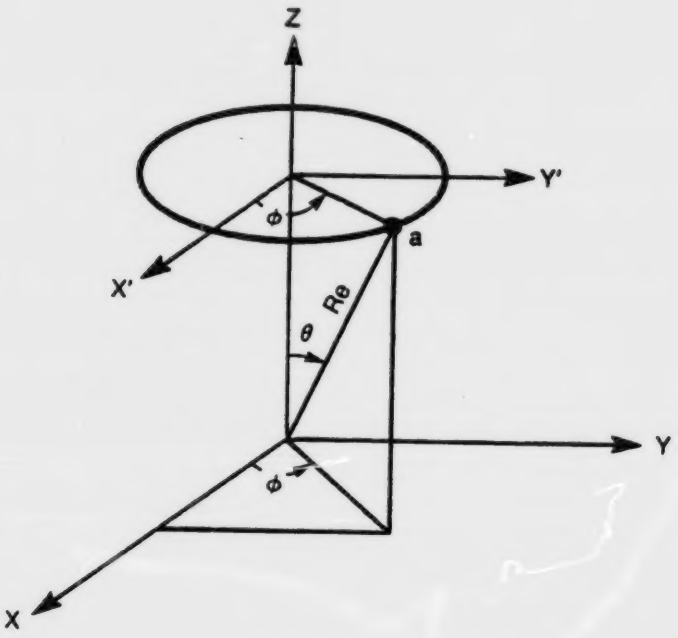


Figure 3a: Circle Defined on an Arbitrary Coordinate System

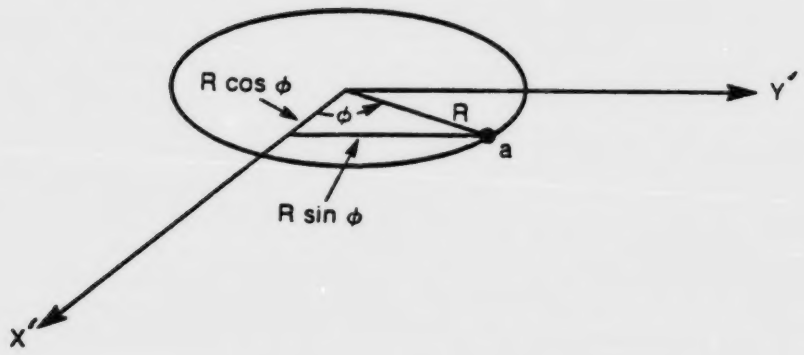


Figure 3b: Top View

NOTE: The X' and Y' axes are superimposed on the circle for reference only. They represent the same axes as the X and Y axes and not another coordinate system.

Referring to figure 3b,

$$y = R \times \sin(\phi)$$
$$= Re \times \sin(\theta) \times \sin(\phi)$$

$$x = R \times \cos(\phi)$$
$$= Re \times \sin(\theta) \times \cos(\phi)$$

The X, Y and Z coordinates of the system that this circle is defined in have to be equated with the Vehicle Centred (VCL) coordinate system shown in Figure A5. This is essential for subsequent coordinate transformations. The axes are matched in the following manner:

X(of VCL) = X(of circle)
Z(of VCL) = Y(of circle)
Y(of VCL) = Z(of circle)

In this manner, the X-axis of the circle is pointing in the direction of the satellite flight, which is the X-axis in the VCL system. The axes are set up in this manner so that the angle of revolution has its zero value at the X-axis. Then, when defining the angle of revolution , for example, 90 degrees, the satellite swath would be 45 degrees on either side of the X-axis, corresponding to 45 degrees on either side of the satellite's line of flight.

The radar's area of illumination is defined in the VCL coordinate system. However, in order to determine whether a target lies within this region, both the area of illumination coordinates and the target coordinates must be in the same reference frame. The Earth Rotating system is used is this model. In order to relate this circle defined in the Vehicle Centred coordinate system to the Earth Rotating system, the following transformations are performed: Vehicle Centred Inertial to Earth Centred Inertial to Earth Rotating to Geodetic Latitude. These transformations are discussed in Appendix A.

2.3 TARGET DETECTION

For the purposes of this report, it is assumed that a target is detected when it is within the viewing cone of the satellite. The two criteria used to determine if the target is detected are: (1) does the target lie between the minimum and maximum grazing angles; and (2) does the target lie within the specified revolution angle of the radar. The method used to determine the changing target position throughout the simulation run is explained in Appendix B.

The method to determine if the target lies between the minimum and maximum grazing angles is described first. The latitude, longitude and height of the target, denoted by TLAT, TLONG and THEIGHT respectively has to be known. Similarly, the latitude, longitude and height of the satellite, denoted by SLAT, SLONG and SHEIGHT respectively, has to be determined. This information for both the target and the satellite have to be obtained for the same instant in time in the orbit in order to be relative to each other. Figure 4 illustrates a vector representation of the target and satellite locations. The grazing angle of the target to the satellite indicates if the target falls within the minimum and maximum grazing angles specified for the study, hence, whether the target can be detected by the satellite. The target's grazing angle is calculated as follows. With reference to Figure 5, define the target's position vector:

$$T(X) = (Re + THEIGHT) \times \cos(TLAT) \times \cos(TLONG)$$
$$T(Y) = (Re + THEIGHT) \times \cos(TLAT) \times \sin(TLONG)$$
$$T(Z) = (Re + THEIGHT) \times \sin(TLAT)$$

A similar procedure is used for the satellite position vector. These two vectors are then transformed to the Inertial Cartesian system using the transformations described in Appendix A. Next, the vector between the satellite and target position vectors is calculated, see Figure 4.

$$D(X) = S(X) - T(X)$$
$$D(Y) = S(Y) - T(Y)$$
$$D(Z) = S(Z) - T(Z)$$

To obtain the grazing angle between the target and the satellite, the dot product between the vectors, T and D is taken, i.e.,

$T \cdot D = ||T|| \times ||D|| \times \cos(\theta)$ where θ is the unknown angle

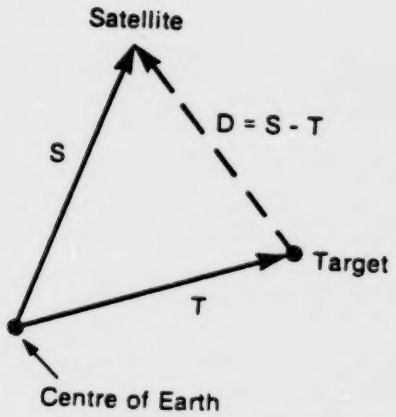


Figure 4a: Vector Representations of Satellite and Target Positions

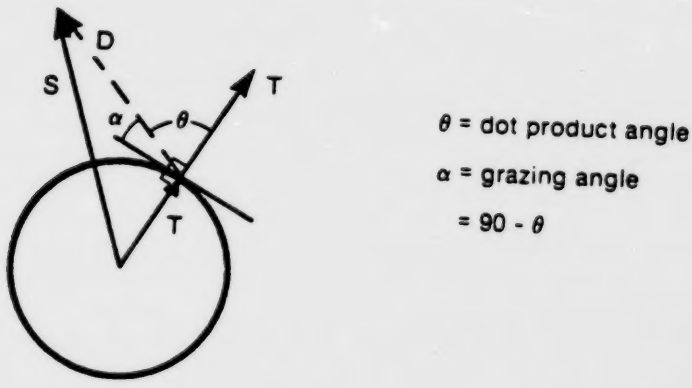


Figure 4b: Vector Representations of Satellite and Target Positions with Respect to the Earth

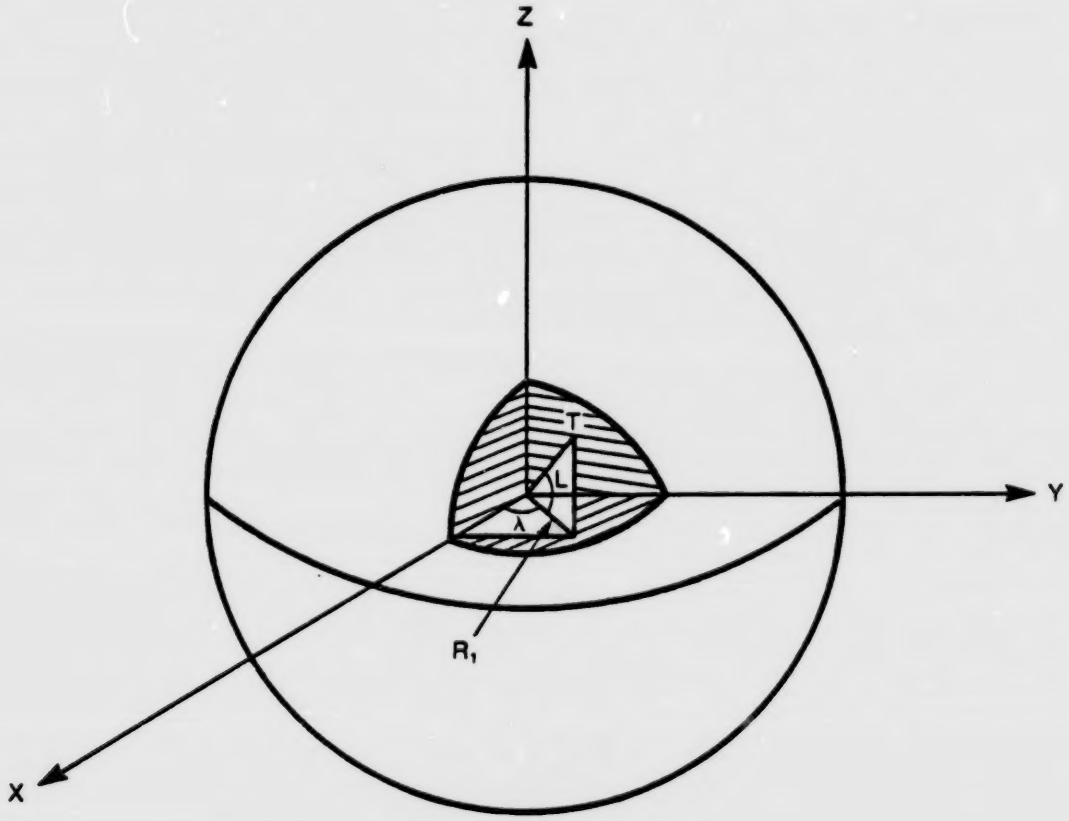


Figure 5: Target Position on a Spherical Earth

Therefore,

$$\cos(\theta) = (T \cdot D) \; / \; (\|T\| \times \|D\|)$$

The target's grazing angle is $90 - \theta = \alpha$.

The above calculation determines if the target is within the 'cone' seen by the satellite. The target must also lie within the specified sector size or angle of revolution of the radar. Figure 6 illustrates the relationship between the satellite and the target on a spherical earth, as well as showing the orbital path of the satellite.

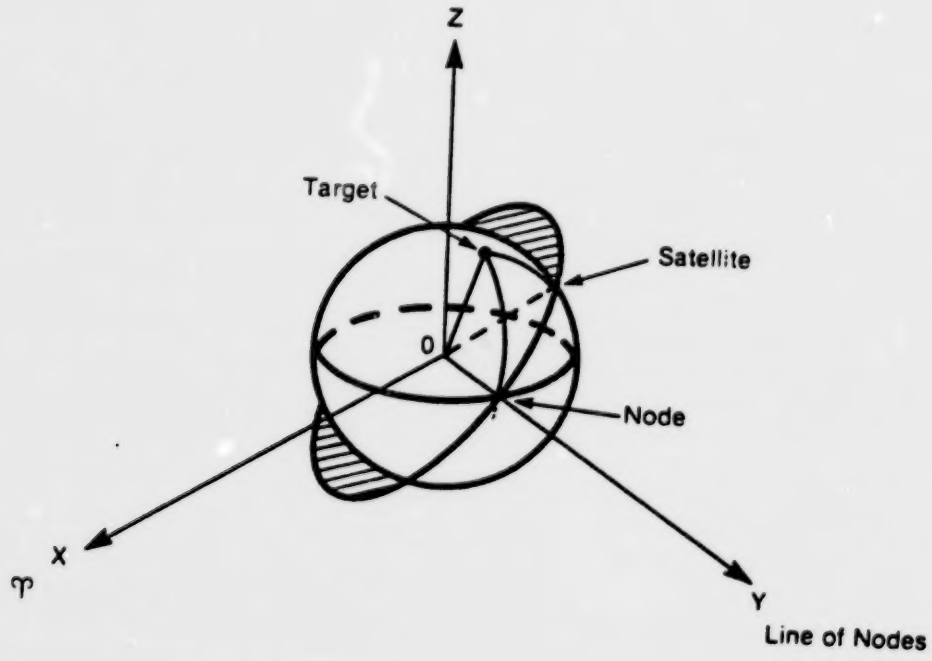


Figure 6: Satellite and Target Positions on a Spherical Earth

To determine if the target lies within the specified revolution angle of the satellite's radar, the satellite and the target position vectors must be transformed from the

Inertial reference frame to the Vehicle Centred reference frame. The inverse of this transformation is detailed in Appendix A. This transformation simplifies the problem since the positive X-axis of the VCL reference frame points in the direction of satellite motion. This approach eliminates the necessity of determining whether the satellite is north bound or south bound. The satellite will always be found on the Z-axis of the VCL frame with zero X and Y components. To determine if the target lies within the specified revolution angle of the satellite's radar, the angle between the target on the X-Y plane and the origin of the VCL coordinate system is calculated. This is done by computing the arctangent(x,y), which returns the angle between the X-axis and the target's position. The target is detected if this angle is within the specified revolution angle.

Both of these criteria described above have to be met in order for the target to be detected by the satellite's radar.

2.4 GRAPHICAL OUTPUT

The overall approach in the analytical development of the model is one of user interaction with the simulation. Therefore, extensive emphasis is placed on computer graphics as a study tool. A brief description of the graphic features found in this model is now provided.

The model implementation was written on a VAX 11/780 using the Tektronix 4100 series of colour graphics terminals. The escape sequences that are utilized in this simulation in order to provide a graphics display are very terminal dependent. Hence, this simulation will only run on the Tektronix 4100 series of terminals. A list of the escape sequences that are used is provided in Appendix E.

The graphic output has been made as dynamic as possible. The terminal used has low dynamic output capabilities and hence is restrictive. With the use of the terminal's capability for segments, dynamic output is possible. This is particularly useful when displaying the satellite's swaths and for target detection. A description of each type of graphic output possible follows.

There are two types of background maps that can be displayed, a polar view of the earth and a mercator view. These are illustrated in Figures 7 and 8 respectively. Both types have their advantages and disadvantages. The polar map accurately depicts the North Pole, the area of most interest to SBR. Distortion of the land masses is apparent near the equator and none of the southern hemisphere is visible.

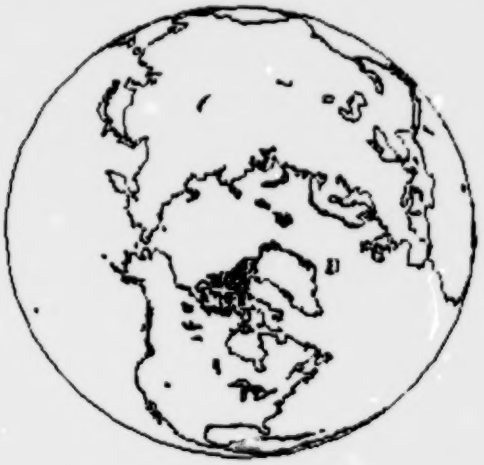


Figure 7: Polar View

The Mercator map displays the entire earth but it distorts the polar regions. These two background maps are used when displaying any of the other graphics outputs.

Figure 8 : Mercator View

It is possible to display the satellite's orbit trace using the polar map. This feature enables the user to visualize the orbit of the satellite around the earth. Figure 9 illustrates an orbit trace.

Figure 9 : Orbit Trace

Ground traces of the satellite's orbit path is also available using either the polar or mercator projection. This feature shows the actual area covered by the satellite taking into account earth rotation, see Figure 10.

Figure 10a: Polar View of Ground Trace

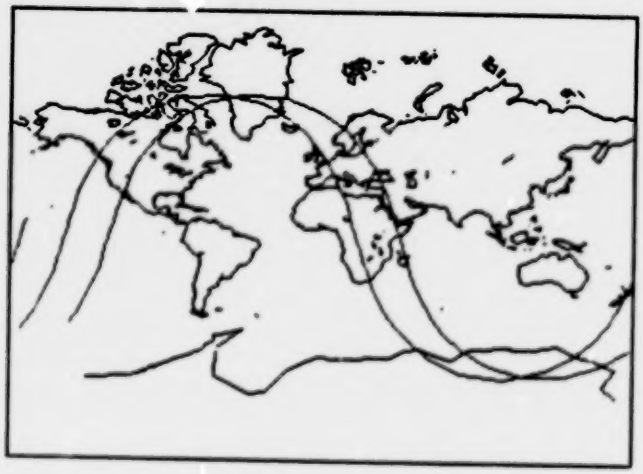


Figure 10b: Mercator View of Ground Trace

Target detection can be shown using either background map. Figure 11 gives a static illustration of the graphic output. This feature provides a 'dynamic' display of the satellite swath and target moving through the simulation time. Upon detection of a target, a 'beep' is heard to alert the user.

Figure 11: Static Display of Target Detection

The mathematical model for a satellite simulation to detect airborne targets has been defined and described. Definitions of orbital parameters have been provided, coordinate systems and coordinate transformations described in Appendix A, the method for determining satellite swaths provided along with the method used in this model for determining if targets are detected. The graphic output features have all been described and illustrated to give an idea of the output available. This model was implemented in order to run various scenarios for testing the performance of target detection.

3.0 IMPLEMENTATION OF MATHEMATICAL MODEL

This section provides a functional description of the software written to implement the model. The purpose of this section is to provide the interested user with an overview of the program modules and how they are interconnected. All the programs and routines are written in Fortran or C for a Vax 11/780. They are written using the escape sequences inherent in the Tektronix 4100 family of terminals. This unfortunately makes the programs very terminal dependant. A program hierarchy is provided for each main module to illustrate the main routines and their subroutines. A users guide is found in Appendix C along with instructions on how to compile and link each of the main routines. Appendix D contains a data dictionary giving details on the main files and arrays used in the routines. Finally, Appendix F documents the listings for all the software written.

The software implementation is comprised of five main routines; DRAWGLOBE, SGPSAT, SWATH, SATPATH and ERASE. These routines are run independently and data is passed between the programs via data files. Each of the main routines is briefly described.

3.1 DRAWGLOBE

The routine, DRAWGLOBE, draws the polar and mercator backdrop maps utilized by the other main programs. This program is a minor modification to an existing globe drawing software package. Details of this package's capabilities can be found in Reference 7. The function of the routine is as follows:

DRAWGLOBE - Main line program that provides run time control. The purpose of this program is to create the backdrop maps for the graphics output from the other routines in the simulation model. Two maps are created, a global polar projection and a mercator projection. Either or both of these maps can be selected.

3.2 SGPSAT

The routine, SGPSAT, simulates satellite motion based upon a set of epoch element conditions as described in section 2.1.1. This program is a conversion and modification of an existing model, see Reference 1 for a complete description of SGPSAT. One of the modifications that was done to this program was to insert the capability to run the program for a constellation of satellites. Four output files are created and they are used as input for the two other main programs in this implementation, SWATH and SATPATH. Figure 12 illustrates the calling sequence hierarchy.

The function of each of these routines is as follows.

3.2.1 General Routines

1. SGPSAT - Main line program that provides run time control. The purpose of this program is to simulate orbiting satellites. This program will generate SGPSAT.DSP and SIMDAT.DAT files which are described later.

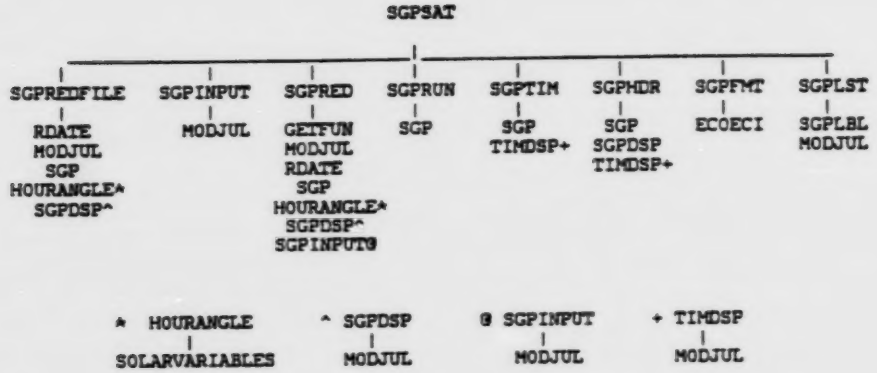


Figure 12: SGPSAT Calling Sequence Hierarchy

2. SGPINS - MACRO to set up PSECT containing the user description of the program

3.2.2 Computational Routines

1. SGP - the primary calculating routine for the satellite simulation. SGP contains all of the mathematical functions that drive the simulation as explained in section 2.1. This routine stores the results in the SGPDAT array, the contents of which are described in Appendix D.

2. SGPTIM - uses the simulation files to find the apogee, perigee, and equatorial crossings. SGPTIM uses search procedures, explained in Reference 1, to perform this task.

3. SGPRUN - runs the simulation by repeated calls to SGP. The FD data array is generated and the output from SGP is stored on a file. For an explanation of the FD array refer to Appendix D.

4. MODJUL - converts dates in standard format (day, month, year, and zulu time) into a Modified Julian date (Number of days since Jan 1,1950) and visa versa.

5. RDATE - takes an ASCII date (eg. 22 APR 65) and converts it into a day, month, year format.

6. ECOECI - converts a vector in Inertial coordinates to orbital coordinates.

7. HOURANGLE - calculates the Greenwich hourangle for a given Modified Julian date.

8. SOLARVARIABLES - a collection of functions that provide information regarding Julian date and orbital parameters of the sun.

3.2.3 Input/Output Routines

1. SGPREDFILE - an input routine that receives epoch parameters for the simulation from the EPOCHDATA.TXT file.

2. SGPRED - a screen editing routine that allows the user to enter the epoch parameters for the simulation at the terminal.

3. SGPINPUT - sets up the screen for the SGPRED screen editor.

4. TITLE - enhances the titles for the SGPRED screen editor.

5. GETFUN - allows the SGPRED screen editor to detect the function keys.

6. SGPDSP - displays the run time elements on the screen.

7. TIMDSP - displays, if requested, the apogee, perigee, and equatorial crossing data that is calculated by SGPTIM.

8. SGPHDR - sets up the output listing headers.

9. SGPFMT - obtains the necessary data for the requested output and stores the data in a file.

10. SGPLST - takes the data files built by SGPFMT and produces formatted display files ready for output to the printer.

11. SGPLBL - organizes the listing page headers for the formatted files created by SGPLST.

There are six different files generated by the SGPSAT program. Four of these files are currently used by this simulation package. The name and logical unit number associated with each file is provided along with a description of the contents of the file.

File name	Logical Unit #	Description
SGPINS.TXT	10	The introductory screen that is displayed on the terminal upon program start up is stored in this file.
SIMDAT.DAT	9	The orbital parameters from the SGPDAT array are written to the file. This file is used to pass the orbital elements input to the other programs in the package.
SGPSIM.DMP	7	All the output requested by the user whether it was written to the screen or a file is stored in this file.
SGPLST.SUM	1	The file contains the display showing the orbital elements and initial conditions.
SGPSIM.DAT	17	The FD array is stored in this file for each computational cycle.
SGPSAT.DSP		This file is comprised of one of four different types of output. The version number of the file depends on which output is specified. Each output type is described separately.
1)Inertial Cartesian	1	The version number is 100 + the run #. The headers describing the contents of the file are as follows:

Date	Time Zulu	T (min)	T Rel (min)	X (Kms)	Y (Kms)	Z (Kms)	VX (M/Sec)	VY (M/Sec)	VZ (M/Sec)

Time (Zulu) provides the simulation time starting at the epoch time given. T(min) gives the simulation time starting at zero and T Rel(min) gives a relative time starting at a time specified by the user. X,Y and Z are the position components of the satellite in the Inertial Cartesian coordinate system. VX, VY and VZ are the velocity components of the satellite in the same coordinate system.

2)Earth Rotating Cartesian	2	The version number is 200 + run #. The headers describing the contents of the file are as follows.

Date	Time Zulu	T (min)	T Rel (min)	X (Kms)	Y (Kms)	Z (Kms)	VX (M/Sec)	VY (M/Sec)	VZ (M/Sec)

The description for these headers are similar to the above Inertial Cartesian headers except that they reference the Earth Rotating Cartesian coordinate system.

3)Geographic Coordinates 3 The version number is 300 + run #. The headers describing the contents of the file are as follows.

Date	Time Zulu	T (min)	T Rel (min)	Longitude (deg east)	Lat(C) (deg)	Lat(G) (deg)	V(L) (M/Sec)	V(E) (M/Sec)	Altitude (Kms)

All the data contained in this file refer to the Geographic coordinate system. Longitude (deg east) indicates the longitude of the satellite in degrees east of Greenwich, Lat(C) gives the Geocentric latitude of the satellite, Lat(G) gives the Geodetic latitude of the satellite, V(L) is the velocity of the satellite in the Inertial Cartesian reference frame, V(E) is the radial velocity of the satellite in the Inertial reference frame, and altitude gives the height of the satellite above the earth's surface.

4)Orbital coordinates 4 The version number is 400 + run #. The headers describing the contents of the file are as follows.

Date	Time Zulu	T (min)	T Rel (min)	True Anomaly (Deg)	R (Kms)	Altitude (Kms)	V(R) (M/Sec)	V(A) (M/Sec)	V(N) (M/Sec)

R is the position vector of the satellite in the Inertial reference frame, V(R) contains the change in the radius vector to the satellite, V(A) contains the change in the direction that the satellite is travelling in and V(N) is the change in the the normal vector between the V(R) and V(A) vectors.

3.3 SWATH

The routine, SWATH, calculates the radar coverage of each satellite. The mathematical concepts for this program are described in section 2.2. The only user interaction with the program is the inputting of the minimum and maximum grazing angles, the revolution angle of the radar and the initial parameters for defining the target's starting position and direction of flight. The latitude and longitude values of the radar coverage for each satellite is output into a file to be used as input into the routine, SATPATH. Figure 13 illustrates the program hierarchy.

The function for each of these routines is as follows:

3.3.1 General Routines

SWATH - Main routine that provides runtime control. The purpose of this program is to calculate the radar coverage swaths of each satellite.

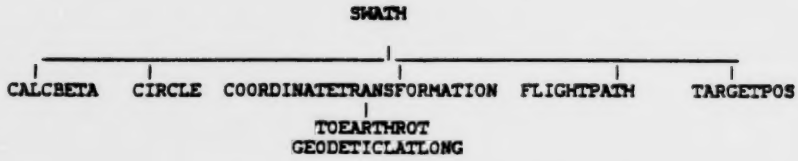


Figure 13: SWATH Calling Sequence Hierarchy

The SWATH program uses the FILEDATA array to produce the GEOCOR.DAT file, which are described in Appendix D.

3.3.2 Computational Routines

1. CALCBETA - computes the beta angles for the minimum and maximum grazing angles of the radar system. An explanation of the beta angles is provided in section 2.2.

2. CIRCLE - calculates the points of the coverage swath for each satellite in the Vehicle Centered coordinate frame. The section 2.2 contains an explanation of the coverage swath.

3. COORDINATETRANSFORMATION - converts the points of the coverage circle from the Vehicle Centered coordinate frame to the Geographic reference frame. These reference frames are explained in Appendix A.

4. TOEARTHROT - a subroutine of COORDINATETRANSFORMATION that converts vectors from the Earth Centered Inertial coordinate frame to the Earth Rotating coordinate frame.

5. GEODETICLATLONG - a subroutine of COORDINATETRANSFORMATION that converts vectors from the Earth Rotating coordinate frame to Geodetic latitude and longitude coordinates.

6. TARGETPOS - computes the target's positions during the time of the satellite orbit. These positions are stored as X, Y, and Z coordinates in the Earth Rotating reference frame.

3.3.3 INPUT/OUTPUT ROUTINES

FLIGHTPATH - obtains the initial parameter values of the target in order to calculate the flight of the target.

There are three files that are used in the program, SWATH. Each file is described separately.

File name	Logical Unit #	Description
SIMDAT.DAT	7	This is the same file as described for the SGPSAT files in section 3.2.3.
SGPSAT.DSP	8 + satellite #	This file contains the orbital coordinate data as described in section 3.2.3. The version number for this file is 400 + satellite number. The satellite number is the same as the run number for the SGPSAT files.
GEOCOR.DAT	8	The first line of this file contains the grazing angles and the sector size input by the user along with the number of satellites in the simulation. The rest of the file contains the latitude and longitude points of the swath circles from the FILEDAT array that is described in Appendix D.

3.4 SATPATH

The routine, SATPATH, generates the graphical output of the simulation data and calculates, then illustrates, whether the target specified is detected. This routine has the ability to illustrate the satellite's orbit path on a polar map, the ground traces on either the polar or mercator map and to illustrate the satellite's swaths on a polar map along with the moving target. Upon detection of the target, a 'beep' will sound indicating to the user that detection has occurred. The principle used for target detection was

described in section 2.3. The calling sequence hierarchy is illustrated in Figure 14 .

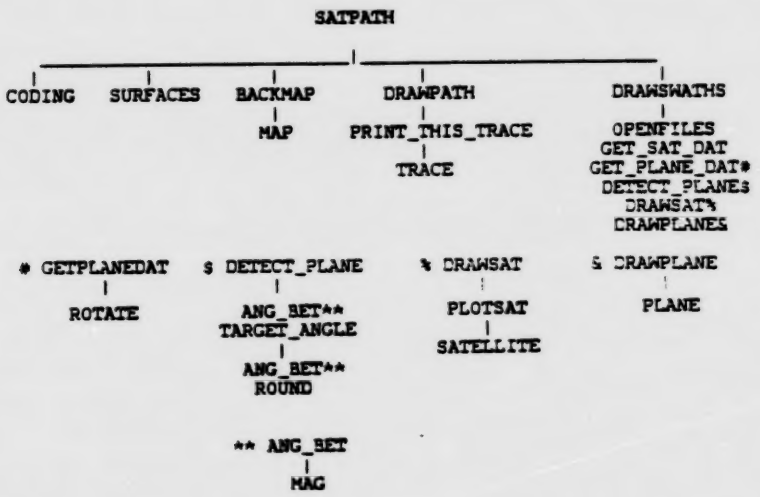


Figure 14: SATPATH Calling Sequence Hierarchy

The function of each of these routines is as follows.

3.4.1 General Routines

1. SATPATH - Main line program that provides run time control. The purpose of this program is to generate graphic output of the simulation data.

2. CODING - converts the responses to the prompts displayed in SATPATH to the following codes: 1 - Mercator projection , 2 - Polar orbits, 3 - Polar ground coverage.

3.4.2 Computational Routines

1. ROTATE - converts a vector from Earth Centered Inertial coordinates to Earth Rotating coordinates.

2. DETECT_PLANE - calculates if the airborne target falls within the swath of a satellite.

3. TARGET_ANGLE - a subroutine of DETECT_PLANE that calculates the angle between the satellite's direction of flight and the airborne target.

4. ANG_BET - calculates the dot product angle between two vectors.

5. MAG - a subroutine of ANG_BET that computes the magnitude of a three dimensional vector.

6. ROUND - rounds off a real number to three decimal places.

3.4.3 Input/Output Routines

1. SURFACES - sets up the Tektronix terminal's surfaces for graphic output.

2. BACKMAP - draws the outline of the map in order to orient the mapping routine.

3. DRAWPATH - draws, if requested, the satellite path data (either the orbit or the ground traces). This routine only draws the unique orbit traces. For example, if more than one satellite share the same orbit, the DRAWPATH routine will only allow this orbit to be drawn once.

4. PRINT_THIS_TRACE - a subroutine of DRAWPATH that opens the proper file for the printing of this particular orbit trace.

5. TRACE - a subroutine of PRINT_THIS_TRACE that actually prints the orbit data on the globe.

6. DRAWSWATHS - produces, if requested, the dynamic simulation of the satellites, their radar swaths, and the airborne target.

7. OPENFILES - opens the proper SGPDSP files as well as reading the Right Ascension and Hourangle data from the SIMDAT.DAT file. This data is stored in a common block for the target detection routines.

8. GET_SAT_DAT - reads the Earth Centered Inertial coordinates of the satellites and stores the data in a common block for use in the plane detection module.

9. GET_PLANE_DAT - generates both the Earth Centered Inertial coordinates of the target and the Earth Rotating coordinates of the airborne target.

10. DRAWSAT - draws the satellites that orbit the polar map.

11. PLOTSAT - a subroutine of DRAWSAT that determines whether each satellite is in front of or behind the earth. If the satellite is in front of the earth, then the satellite is drawn.

12. SATELLITE - contains the data needed to draw the actual satellite.

13. DRAWPLANE - draws the airborne target.

14. PLANE - contains the data needed to draw the airborne target.

There are four files that are used in the program, SATPATH. Each file is described separately.

File name	Logical Unit #	Description
GEOCOR.DAT	8	Identical file to the one described in section 3.3.3.
SGPSAT.DSP	8 + satellite #	These files contain Earth Rotating coordinate system data as described in section 3.2.3. The version numbers are 200 + satellite number.
	23 + satellite #	These files contain Inertial Cartesian coordinate system data as described in section 3.2.3. The version numbers of these files are 100 + satellite number.
SIMDAT.DAT	39	This is the same file as was described above for the SGPSAT file in section 3.2.2.
DETECTPOS.DAT	21	This file contains the satellite number, time and location of the airborne target at the points of detection.

3.5 ERASE

The routine, ERASE, clears the screen of all graphics that has been displayed after running SATPATH. It facilitates the erasing of the different surfaces that have been used in the Tektronix 4107 terminal.

A User's Guide explaining how to compile and link each of the four main programs is found in Appendix C along with how to run each of the programs. Appendix D contains a data dictionary of the main files and arrays used in this simulation package.

4.0 SUMMARY DISCUSSION

A mathematical model for the detection of airborne targets by radars on satellites has been described. A brief explanation of the pertinent orbital parameters was provided along with a description of the coordinate systems and coordinate transformations that were utilized. The method of target detection by the radar was also explained and a brief description of the software created to implement this model was given. An explanation of the method used to update the airborne target position during the simulation has also been provided. Upgrading of this model is necessary, especially in the areas of radar and antenna concepts.

Several areas have been identified that require future development. These areas are: (1) a radar model, (2) a detection model and (3) the inclusion of several airborne target profiles. Presently, the target is 'detected' when it is within the bounds defined by the radar's area of illumination. There needs to be a capability to alter the radar and antenna parameters to determine the effect on target detection with several types of radars and antennas. More development is also needed in the area of scanning the radar's footprint for the airborne target. Varying the scanning algorithm will permit the study of its effect on target detection. With these future developments incorporated into the simulation model, a comprehensive system tool will have been created for systems studies.

concept design work, probability and trade off studies.

REFERENCES

1. "The Sat Program, A satellite simulation program", W. R. McPherson, March 1980. Report No. IGDS 1/80.

2. "Methods of Celestial Mechanics", D. Brouwer, G. M. Clemence, Academic Press, Inc., New York, 1961

3. "Introduction to Celestial Mechanics", J. Kovalevsky, Springer-Verlag New York Inc., New York, 1967

4. "Fundamentals of Astrodynamics", R. R. Bate, D. D. Mueller, J. E. White, Dover Publications, New York, 1971

5. "The Motion of a Close Earth Satellite", Astronomical Journal, Y. Kozai, 1959

6. "Note on the Motion of a Close Earth Satellite with a Small Eccentricity", Astronomical Journal, Y. Kozai, 1961

7. "'GLOBE' - A Library of Map Generating Subroutines", Dirk A. Bell, 1980

8. "A Perturbation Model for Low Earth Orbit Satellites", M.W. Lovell, S.Y. Slinn, W.R. McPherson, DREO Technical Note 86-

Appendix A

Coordinate Systems and Coordinate Transformations

A.1 COORDINATE SYSTEMS

In order to describe an orbit, a suitable inertial reference frame is needed. Several coordinate systems are used in this model. Each will be described giving the position of the origin, the orientation of the fundamental plane (the X-Y plane), the principal direction (the direction of the X-axis), and the direction of the Z-axis. The five coordinate systems that are utilised are the Inertial Cartesian, Earth Rotating, Geographic, Orbital, and Vehicle Centred.

The Inertial Cartesian (or Geocentric-Equatorial) reference frame has the fundamental plane in the equatorial plane and its origin at the earth's centre. The principal direction of this system points towards the Vernal Equinox direction and the Z-axis points in the direction of the north pole. This system is nonrotating with respect to the stars, hence, it is not fixed to the earth or rotating with it. The Inertial Cartesian reference frame is illustrated in Figure A1.

The Earth Rotating reference frame is similar to the Inertial system except that the principal direction points in the direction of the prime meridian. Thus, this coordinate system rotates at the earth rotational rate. Figure A2 illustrates the Earth Rotating system.

The Geographic reference frame is similar to the Earth Rotating system except that the location of an object is measured in polar coordinates. The vector coordinates of the object's location is: R - the radial distance to the object being measured, L - the latitude, positive north of the equator and λ - longitude, east of the prime meridian. Figure A3 illustrates the Geographic coordinate system.

The Orbital (or Perifocal) reference frame has its fundamental plane as the satellite's orbital plane. This makes it a convenient system for describing the motion of a

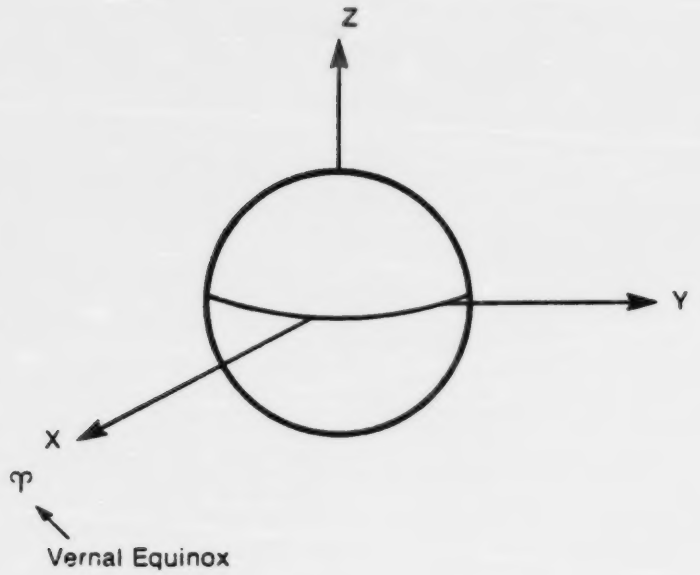


Figure A1: Inertial Cartesian Reference Frame

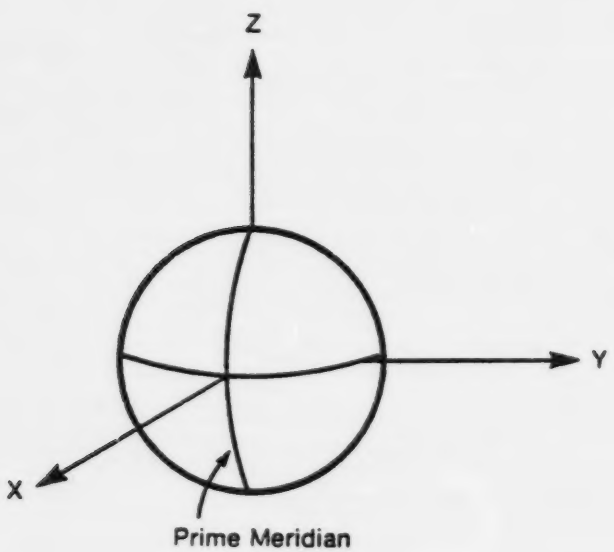


Figure A2: Earth Rotating Reference Frame

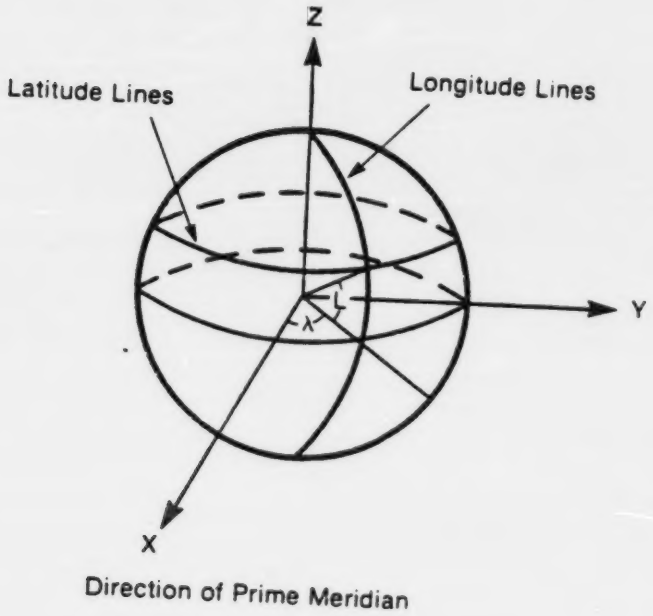


Figure A3: Geographic Reference Frame

satellite. The fundamental direction points towards perigee and the Z-axis is normal to the orbital plane. This system is illustrated in Figure A4.

The Vehicle Centred reference frame has its fundamental plane as the satellite's orbital plane. It's origin is centred at the satellite. The fundamental direction points towards the satellite's direction of motion and the Z-axis is normal to the orbital plane. Figure A5 illustrates this system.

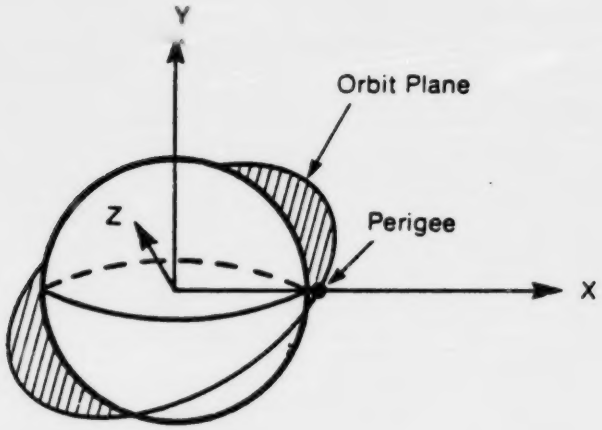


Figure A4: Orbit Reference Frame

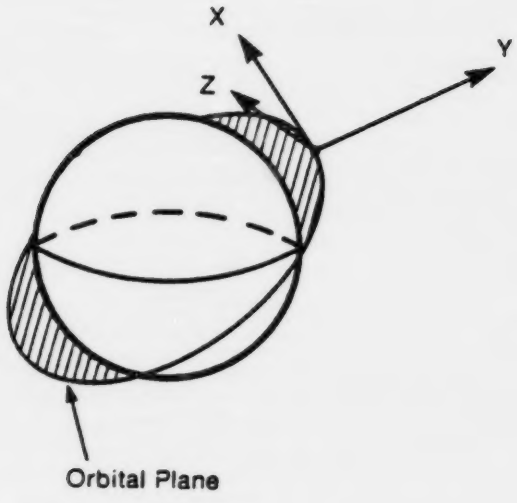


Figure A5: Vehicle Centred Reference Frame

A.2 COORDINATE TRANSFORMATIONS

The coordinate system that the satellite is defined in is frequently altered throughout the model. Some calculations are performed more 'naturally' in one coordinate system but then have to be transformed to ensure that subsequent calculations are all performed referencing the same system. A basic review of coordinate transformations is presented along with the transformation matrices that were developed for use in the model.

A coordinate transformation changes the basis of a vector. A "basis" is a collection of unit vectors indicating the direction of the three perpendicular axes in a coordinate frame. The vector has the same properties and identical representation after the transformation. Positive rotations about any axis is defined by the 'right hand rule'. If the thumb of the right hand is extended in the direction of the positive coordinate axis, the fingers curl in the direction of a positive rotation. The order in which the transformations are performed is important due to the fact that different results are obtained when matrices are multiplied in different orders.

All transformations perform a rotation about one or more of the three axes. There are two methods of performing a rotation: (1) rotation of a vector about the axes or (2) rotation of the axes. Different transformation matrices are utilized depending on what type of rotation is being performed. Throughout the development of this model, both types are used. An explanation of the development of transformation matrices for both types of rotations is provided in the section A.3. The transformation matrix for a single rotation of each of the three axis is as follows:

Rotation of the positive X-axis through a positive angle α.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

The coordinates of a position transformed about this axis is determined by,

$$\begin{bmatrix} Xnew \\ Ynew \\ Znew \end{bmatrix} = A \times \begin{bmatrix} Xold \\ Yold \\ Zold \end{bmatrix}$$

Rotation of the positive Y-axis through a positive angle α .

$$B = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

Rotation of the positive Z-axis through a positive angle α .

$$C = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Coordinate transformations from one system to another generally involve more than one rotation. The satellite simulation model requires four different coordinate transformations: Orbital to Inertial Cartesian, Orbital to Earth Rotating, Geocentric to Geodetic Latitude and Vehicle Centred to Inertial Cartesian. Each will be discussed in detail.

A.2.1 ORBITAL TO INERTIAL CARTESIAN COORDINATES

The relationship between the Orbital and Inertial Cartesian coordinate systems is defined by the three angles:

ω - argument of perigee
i - inclination
Ω - longitude of ascending node

Figure A6 illustrates the relationship between these two systems. The Orbital system is represented by the P,Q,W unit vectors, and the Inertial Cartesian system is represented by the I,J,K unit vectors.

The first rotation is about the W-axis. A rotation of $-\omega$ will cause the P-axis to rotate to the line of nodes. The transformation matix for this operation is:

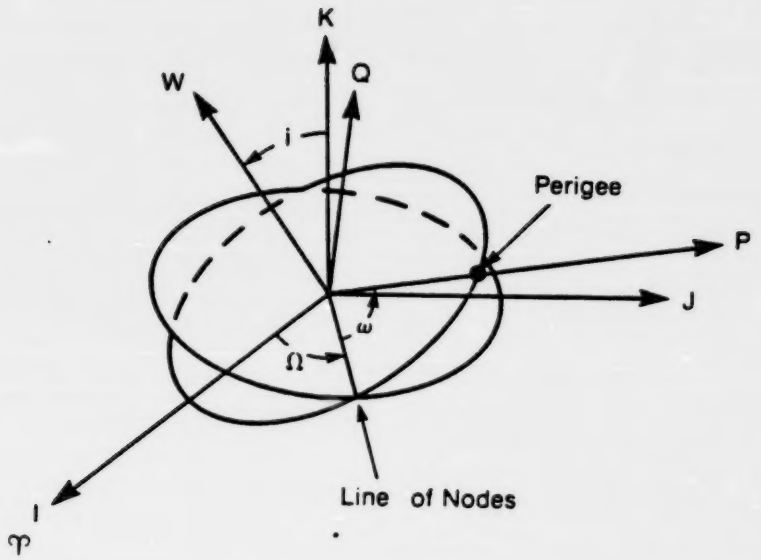


Figure A6: Relationship between the Inertial Cartesian and Orbital Reference Frames

$$A = \begin{bmatrix} \cos(\omega) & -\sin(\omega) & 0 \\ \sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The next rotation involves the newly relocated P-axis. A rotation of -i causes the orbital plane to coincide with the equatorial plane. The transformation matrix for this operation is:

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & -\sin(i) \\ 0 & \sin(i) & \cos(i) \end{bmatrix}$$

The W-axis now lies on the K-axis of the Inertial Cartesian system. The last rotation is about the newly relocated W-axis in order to bring the P-axis to the I-axis at the Vernal Equinox. A rotation of - Ω about the new W-axis will perform this transformation. The matrix for this operation is:

$$C = \begin{bmatrix} \cos(\Omega) & -\sin(\Omega) & 0 \\ \sin(\Omega) & \cos(\Omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiply these three matrics together, A, B and C, to produce the transformation matrix D, for an Orbital to Inertial Cartesion Coordinate transformation.

Let:

$cw = \cos(\omega)$ $\quad sw = \sin(\omega)$
$ca = \cos(\Omega)$ $\quad sa = \sin(\Omega)$
$ci = \cos(i)$ $\quad si = \sin(i)$

$$D = C \times B \times A =$$

$$\begin{bmatrix} (cw)(ca)-(sw)(ci)(sa) & (-sw)(ca)-(cw)(ci)(sa) & (si)(sa) \\ (cw)(sa)+(sw)(ci)(ca) & (-sw)(sa)+(cw)(ci)(ca) & (-si)(ca) \\ (sw)(si) & (cw)(si) & (ci) \end{bmatrix}$$

To perform the inverse transformation, Inertial Cartesian to Orbital coordinate transformation, the transpose of the matrix D is taken. This will result in it's inverse matrix.

A.2.2 INERTIAL CARTESIAN TO EARTH ROTATING COORDINATES

Another coordinate transformation used is the Inertial Cartesian to Earth Rotating coordinate system. The relationship between these two coordinate systems is defined by the hour angle. Hour angle is the angle between the X-axis in the Inertial Cartesian system and the X-axis in the Earth Rotating system, see Figure A7. This angle provides the location of the longitude of Greenwich from the Vernal Equinox direction at epoch.

The only rotation required for this transformation is a positive rotation of the hour angle about the Z-axis. The matrix for this operation is:

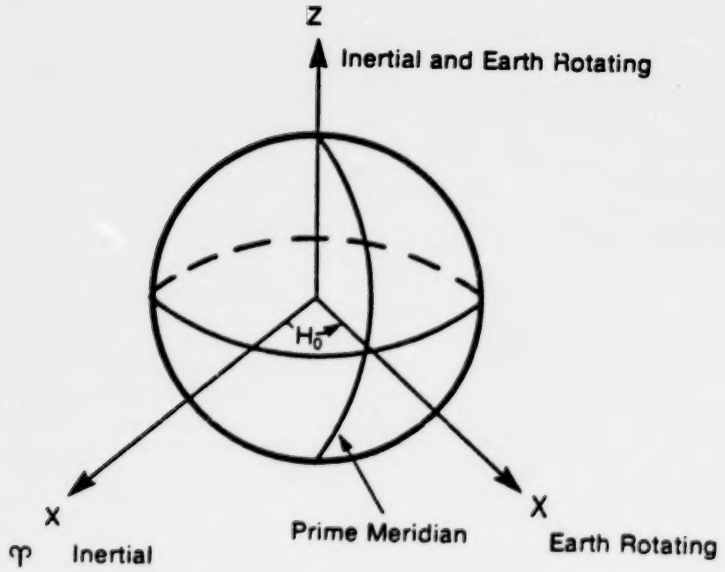


Figure A7: Hour Angle

$$A = \begin{bmatrix} \cos(H_o) & \sin(H_o) & 0 \\ -\sin(H_o) & \cos(H_o) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A.2.3 GEOCENTRIC TO GEODETIC COORDINATES

The transformation from geocentric to geodetic coordinates is essential because the earth is not spherical, it is slightly flattened at the poles. All the calculations in the model are performed assuming a spherical earth, but for accurate results, they have to be transformed to geodetic coordinates. The earth's equatorial radius is greater than the polar radius. Since the radius is a function of latitude, the latitude can not be represented in spherical coordinates. An ellipsoid earth poses no problems on the measurement of longitude. The difference in 'spherical' and 'ellipsoid' latitudes is illustrated in Figure A8. The angle α represents the "geocentric"

latitude. This latitude is defined to be the angle between the equatorial plane and the radius from the geocentre. The angle ϕ represents the "geodetic" latitude. This latitude is defined to be the angle between the equatorial plane and the normal to the surface of the ellipsoid.

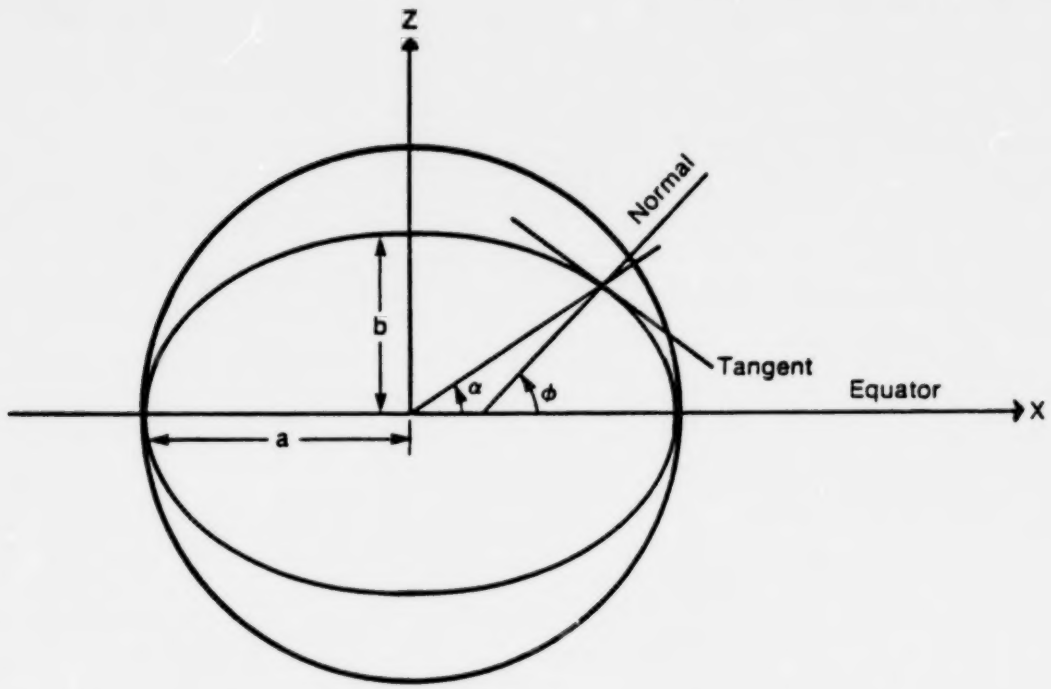


Figure A8: Geocentric vs. Geodetic Latitude

To transform geodetic latitude to geocentric latitude, the following equation is used:

$$\tan(\text{geodetic latitude}) = \tan(\text{geocentric latitude})/(1-f^2)$$

where f = the earth flattening, 1/298.25

The earth's radius at any latitude is:

$$R = AE \times (1-f)/\sqrt{(1-(2f - f^2) \times \cos(\text{geocentric latitude}))}$$

where AE = the mean equatorial radius, 6378.145 km.

A.2.4 CARTESIAN TO SPHERICAL COORDINATES

Figure A9 represents an Earth Cartesian coordinate system.

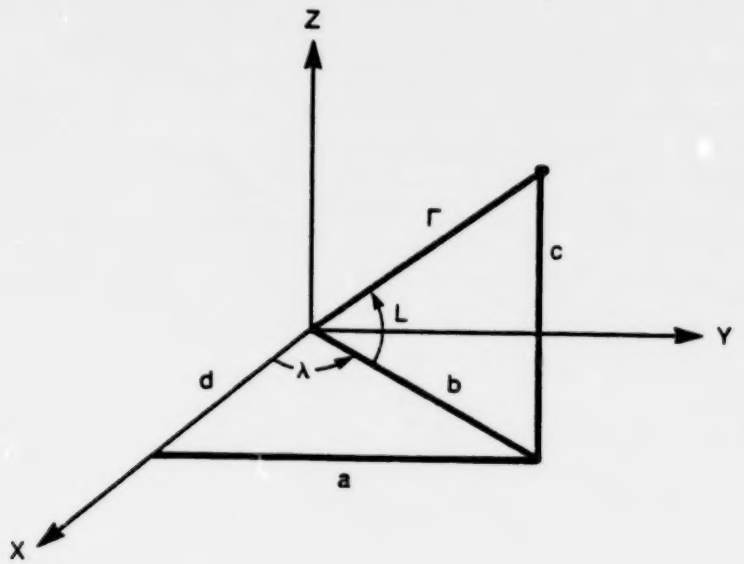


Figure A9: Earth Cartesian Coordinate System

The transformation from Cartesian to Spherical coordinates is as follows. Let Rx, Ry, Rz represent the Earth Cartesian coordinates, λ the longitude east, and L the latitude. It then follows that:

$$\tan(\lambda \text{ geocentric}) = Ry/Rx$$

$$\tan(L \text{ geocentric}) = Rz/\sqrt{(Rx^2) + (Ry^2)}$$

$$\text{geocentric radius} = \sqrt{(Rx^2) + (Ry^2) + (Rz^2)}$$

The converse transformation, Spherical to Earth Cartesian coordinates is more involved. Each of the Cartesian coordinates will be dealt with separately, again with reference to Figure A9.

$$\text{let } r = \text{geocentric radius of the sphere} = \sqrt{(Rx^2) + (Ry^2) + (Rz^2)}$$

Rx: $\cos(\lambda) = d/b = Rx/\sqrt{(Rx^2) + (Ry^2)}$

Rearrange this equation, to solve for Rx,

$$Rx = \cos(\lambda) \times \sqrt{(Rx^2) + (Ry^2)}$$

but

$$\cos(L) = b/r = \sqrt{(Rx^2) + (Ry^2)}/r$$

Therefore,

$$Rx = \cos(\lambda) \times \cos(L) \times r$$

Ry: $\tan(\lambda) = a/d = Ry/Rx$

$$Ry = Rx \times \tan(\lambda)$$
$$= Rx \times (\sin(\lambda)/\cos(\lambda))$$

but

$$\cos(\lambda) = d/b = Rx/\sqrt{(Rx^2) + (Ry^2)}$$

Therefore,

$$Ry = Rx \times \sin(\lambda) \times (\sqrt{(Rx^2) + (Ry^2)}/Rx)$$
$$= \sin(\lambda) \times \sqrt{(Rx^2) + (Ry^2)}$$

but

$$\cos(L) = b/r = \sqrt{(Rx^2) + (Ry^2)}/r$$

Therefore,

$$Ry = \sin(\lambda) \times \cos(L) \times r$$

Rz: $\sin(L) = c/r = Rz/r$

Therefore,

$$Rz = \sin(L) \times r$$

To summarize the above equations, the transformation matrix for Spherical to Earth Cartesian coordinates is:

$$\begin{bmatrix} Rx \\ Ry \\ Rz \end{bmatrix} = r_{geocentric} \begin{bmatrix} \cos(L) \times \cos(\lambda) \\ \cos(L) \times \sin(\lambda) \\ \sin(L) \end{bmatrix}$$

where L is the geocentric latitude and λ is the geocentric longitude.

A.2.5 VEHICLE CENTRED TO EARTH CENTRED INERTIAL COORDINATES

The last transformation to be discussed is the Vehicle Centred coordinate system to the Earth Centred Inertial coordinate system.

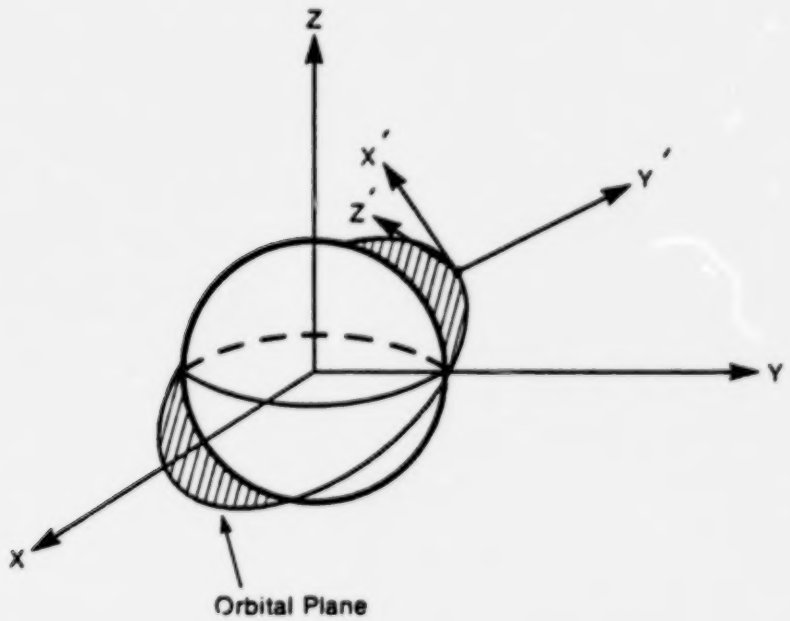


Figure A10: Vehicle Centred and Earth Centred Inertial Coordinate System

Figure A10 illustrates both the Vehicle Centred axes, X', Y', and Z', and the Earth Centred Inertial axes, X, Y, and Z. For this transformation, the following symbols are used:

ω - argument of perigee
θ - true anomaly
i - inclination
n - longitude of ascending node

The first rotation is about the Z'-axis by an angle of $-(\theta + \omega)$. This causes the Y'-axis of the Vehicle Centred coordinate sytem to coincide with the line of nodes. The transformation matrix for this operation is:

$$A = \begin{bmatrix} \cos(\theta+\omega) & -\sin(\theta+\omega) & 0 \\ \sin(\theta+\omega) & \cos(\theta+\omega) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Next, a rotation about the Y'-axis by -i brings the X'-axis in line with the Earth Centred Inertial Y-axis. The transformation matrix is:

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & -\sin(i) \\ 0 & \sin(i) & \cos(i) \end{bmatrix}$$

A, rotatation about the Z'-axis by (-n) brings the Y'-axis from the line of nodes to point to Vernal Equinox. The transformation matrix is:

$$C = \begin{bmatrix} \cos(n) & -\sin(n) & 0 \\ \sin(n) & \cos(n) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying these three matrices together gives the transformation matrix for a Vehicle Centred coordinate system to a Earth Cartesian Inertial coordinate system.

Let:

co = cos(n) so = sin(n)
ci = cos(i) si = sin(i)
cw = cos(θ + ω) sw = sin(θ + ω)

D = C x B x A =

$$\begin{bmatrix} (co)(cw)-(so)(ci)(sw) & -(co)(sw)-(so)(ci)(cw) & (so)(si) \\ (so)(cw)+(co)(ci)(sw) & -(so)(sw)+(co)(ci)(cw) & -(co)(si) \\ (si)(sw) & (si)(cw) & (ci) \end{bmatrix}$$

In order to perform the inverse transformation, Earth Cartesian Inertial to Vehicle Centred , take the transpose of the matrix D. This will result in it's inverse matrix.

The satellite's orbit, described by the Classical set of orbital elements, is represented in several coordinate systems throughout the model. Many transformations are performed between the coordinate systems as some calculations are more easily performed in one system.

A.3 ROTATION OF COORDINATE AXES VS. ROTATION OF VECTORS.

Two types of coordinate transformations are possible: (1) a rotation of a vector about the axes and (2) a rotation of the axes. Different transformation matrices are utilized depending on what type of rotation is being performed. This appendix provides details on the development of each type of transformation matrix.

A vector V is transformed into a vector V' under the effect of a rotation ie:

$$V \xrightarrow{(RA)} V'$$

In matrix notation,

$$[RA]\, V = V'$$

with

$$V' = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \quad \text{and} \quad V = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = X \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + Y \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + Z \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Due to the linearity of the rotation matrix, this can be written as:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = X [RA] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + Y [RA] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + Z [RA] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

First, a rotation of the axes is discussed. For illustration purposes, a rotation of the axes about the Y-axis is developed. A rotation about this axis will create new X and Z axes as illustrated in Figure A11. The points A and B are the endpoints of basis vectors found on the original X and Z axes. After the rotation, these vectors are in the same position, it is the X and Z axes that have changed location. The transformation matrix for a rotation of the axes is as follows:

$$[RA] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ 0 \\ \sin(\theta) \end{bmatrix}$$

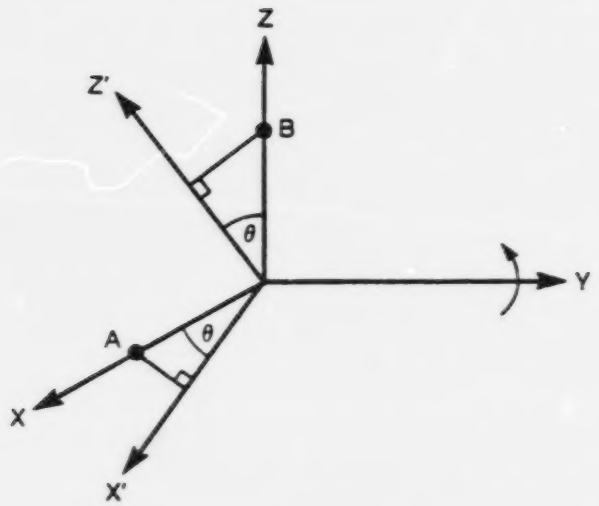


Figure A11: Rotation of Axes about the Y-Axis

and

$$[RA]\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

and

$$[RA]\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}-\sin(\theta)\\0\\\cos(\theta)\end{bmatrix}$$

Hence,

$$RA(\theta) = \begin{bmatrix}\cos(\theta) & 0 & -\sin(\theta)\\0 & 1 & 0\\\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}$$

Figure A12 illustrates the rotation of vectors around the Y-axis.

Assume that the points A and B are the endpoints of the new basis vectors. For this transformation, the location of the axes do not change, the location of the vectors change. The coordinate transformation for this rotation is:

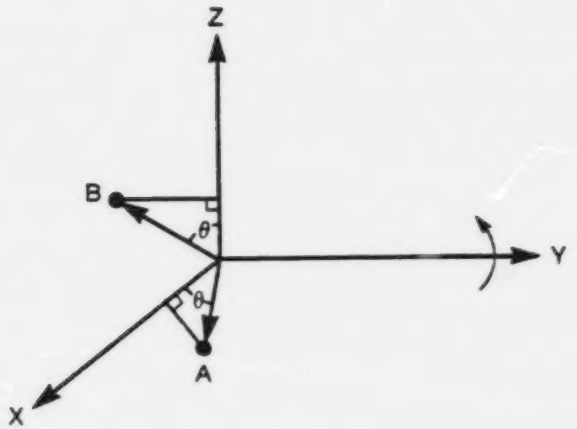


Figure A12: Rotation of Vectors around the Y-Axis

$$[RV]\begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}\cos(\theta)\\0\\-\sin(\theta)\end{bmatrix}$$

and

$$[RV]\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

and

$$[RV]\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}\sin(\theta)\\0\\\cos(\theta)\end{bmatrix}$$

Therefore,

$$RV(\theta) = \begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\0 & 1 & 0\\-\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}$$

The array RA is the transpose of the array RV. Therefore, the determination of which type of rotation is being performed is crucial as different outcomes will result depending upon which array is utilised.

APPENDIX B

B.1 METHOD OF TARGET POSITION DETERMINATION

This appendix provides a discussion on the method used in the simulation model for target motion. Future enhancements are needed in this area to make the target's motion more realistic.

The user inputs the values of the target's starting latitude and longitude, its original bearing, and its velocity when prompted by the program SWATH. From these initial parameters, the target position is calculated at specified time intervals during the run. The calculations are performed using the following assumptions; (1) the target's velocity is constant, (2) the target does not change its course, (3) the altitude of the target is negligible and (4) the earth is spherical.

Any two distinct points on a sphere (which are not the opposite ends of a diameter) lie on one and only one great circle. The shortest distance joining the two points is an arc of this great circle. A spherical triangle is constructed by connecting the initial location of the airborne target (A) and its position at the end of the time interval (B) to the North Pole by arcs of their meridians as shown in Figure B1. The third side of the triangle is the path travelled from A to B. Using spherical geometry, the coordinates of the target's new location are determined.

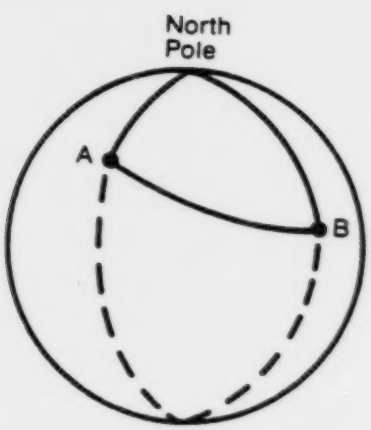


Figure B1: Constructing a Spherical Triangle

The target's bearing, labelled α in Figure B2, is the angle, measured clockwise from north, between the north line through the target's initial position and it's flightpath. Arc b, in Figure B2, represents the co-latitude of A and is calculated by subtracting the starting latitude of the target from 90 degrees. Arc p is the departure of point B from point A and its length is computed as follows:

$$p = \text{velocity of target} \times \text{delta time}$$

Since p is the arc of a great circle, 1 nautical mile = 1 minute, and the arc length can be converted to minutes or to degrees. From these values (α,b,p) the difference in longitude between the two positions and the co-latitude of B (arc length a) are found, which in turn specifys the co-ordinates of B.

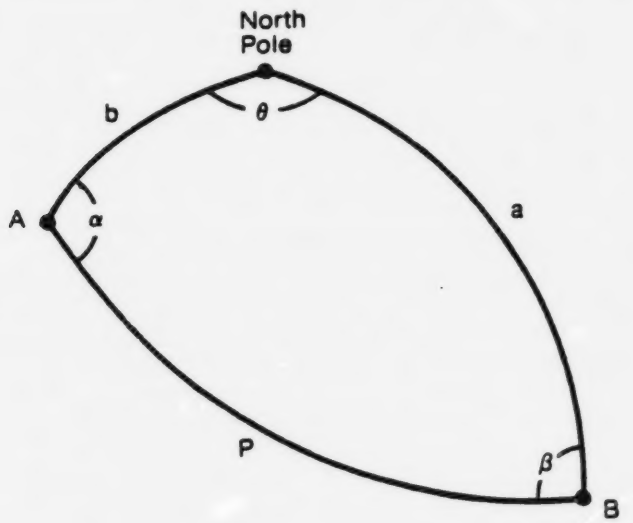


Figure B2: Enlarged View of the Spherical Triangle

Quadrant checking is eliminated by using the haversine method of solving spherical triangles. The haversine of an angle is defined as follows:

$$\text{hav}(\alpha) = 0.5 \times (1 - \cos(\alpha))$$

Then,

$$hav(\theta) = 0.5 \times (1 - \cos(\theta))$$

from the law of cosines for sides of a spherical triangle:

$$\cos(\theta) = (\cos(p) - \cos(a) \times \cos(b)) / (\sin(a) \times \sin(b))$$

so,

$$\begin{aligned} hav(\theta) &= 0.5x((\sin(a)x\sin(b)+\cos(a)x\cos(b)-\cos(p))/(\sin(a)x\sin(b))) \\ &= 0.5x((\cos(a-b)-\cos(p))/(\sin(a)x\sin(b))) \\ &= (-\sin \tfrac{1}{2}(a-b+p) \times \sin \tfrac{1}{2}(a-b-p)) / (\sin(a) \times \sin(b)) \end{aligned}$$

substituting $(a+b+p)/2 = s$

$$\begin{aligned} hav(\theta) &= \sin(s-c) \times \sin(s-b) \times \csc(a) \times \csc(b) \\ \theta &= \arccos(1 - 2 \times \sin(s-b) \times \sin(s-p) \times \csc(a) \times \csc(b)) \end{aligned}$$

Also,

$$\begin{aligned} hav(a) &= 0.5x(1-\cos(b)x\cos(c)-\sin(b)x\sin(c)x\cos(A)) \\ &= 0.5x(1-\cos(b)x\cos(c)-\sin(b)x\sin(c)x(1-2xhav(A))) \\ &= 0.5x(1-\cos(b)x\cos(c)-\sin(b)x\sin(c)+2xhav(A)x\sin(b)x\sin(c)) \\ &= 0.5x(1-\cos(b-c)+2x\sin(b)x\sin(c)xhav(A)) \end{aligned}$$

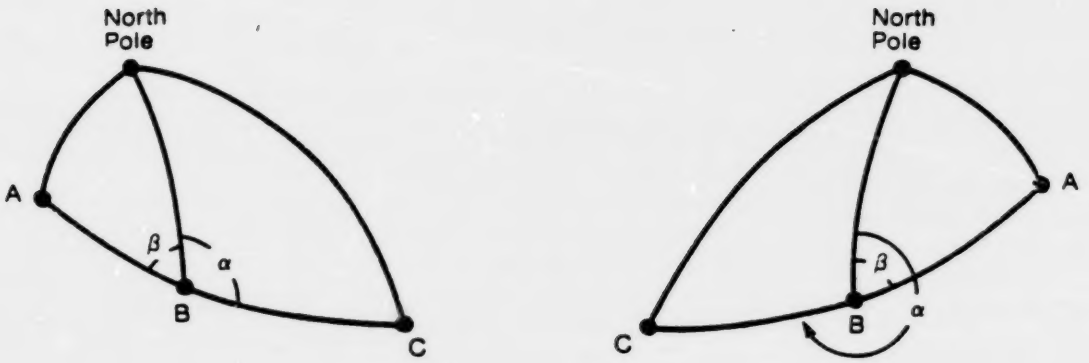
Then,

$$\begin{aligned} hav(a) &= hav(b-c) + \sin(b) \times \sin(c) \times hav(A) \\ a &= \arccos(1 - 2 \times (hav(b-c) + \sin(b) \times \sin(c) \times hav(A))) \end{aligned}$$

For travel in an easterly direction (bearing $< 180°$), the value of θ is added to the starting longitude to find the present longitude of the target. The difference is subtracted for travel in a westerly direction (bearing $> 180°$). The present latitude is the remainder when 'a' is subtracted from 90 degrees. The new bearing of the target, , is found by subtracting angle β from 180 degrees for travel in an easterly direction or adding β to 180 degrees for travel to the west; refer to Figure B3. Angle β is found using the equation:

$$\beta = \arccos(1 - 2 \times \sin(s-p) \times \sin(s-a) \times \csc(a) \times \csc(p))$$

The target's latitude, longitude and bearing are updated and the target's next location can be found.



Travel in an
Easterly Direction

Travel in a
Westerly Direction

Figure B3: Target Bearing

APPENDIX C

Users Guide

C.1 COMPILATION AND LINKING

This section of the appendix describes how to compile and link the five main programs and their related subroutines for the simulation model.

All the modules, except GRAPHICS1, HOURANGLE, SOLARVARIABLES and SCREEN, have been written in FORTRAN. The other four were written in C. The command to compile a FORTRAN module, eg. MODULE_NAME.FOR, is:

```
$ F MODULENAME
```

The command to compile a C module, eg. MODULE_NAME.C, is:

```
$ CC MODULENAME
```

In order to link several modules together, eg: MODULE_NAME1, MODULE_NAME2, and MODULE_NAME3, compile each one as illustrated above, then link the object modules together with the command:

```
$ LINK MODULENAME1,MODULENAME2,MODULENAME3
```

A description on the routines to compile and link for each of the five main programs follows.

For the DRAWGLOBE program the modules to compile are:

DRAWGLOBE, GRAPHICS, GRAPHICS1

These must be linked with the PLOT10 library,

```
[PLOTTER]PLOT10/LIB
```

The linking process has been simplified by the use of a command procedure which is invoked by the command:

$@MAPLINK

For the SGPSAT program the modules to compile and link are:

SGPSAT, SOLARVARIABLES, BLOCKSGP, SGPRED, MODJUL, SGP, SGPDSP, SGPTIM, TIMDSP, RDATE, SGPRUN, SGPHDR, SGPFMT, ECOECI, SGPLST, SGPLBL, GRAPHICS, GRAPHICS1, SGPREDFILE, HOURANGLE, SCREEN, SGPINPUT, FSCREEN

Invoke the command procedure to link these object modules with the command:

$@SGPLINK

The SWATH program has only one module to be compiled and linked. The commands to do these functions are:

```
$ F SWATH
$ LINK SWATH
```

The SATPATH program, like the SGPSAT program, has many modules to compile and link. These modules are:

SATPATH, DRAWSWATH, MAPPER, BACKMAP, GRAPHICS, GRAPHICS1, DRAWPATH, DRAWSAT, SURFACES, GETSATDAT, DRAWPLANE, DETECT, OPENFILES, CALCULATEANGLE, ARCTANGENT2

These must be linked with the globe drawing routine and the PLOT10 library,

MAPPER and [PLOTTER]PLOT10/LIB

Invoke the linking process with the command:

$@GLINK

The ERASE program has only one module to be compiled and linked. The commands to do these functions are:

```
$F ERASE
$LINK ERASE, GRAPHICS
```

A listing of all the command procedures described above is provided in Appendix F.

C.2 RUNNING THE PROGRAMS

A program can be run only after it's modules have been compiled and linked. To run this simulation, the four main programs have to be invoked in the following order, DRAWGLOBE, SGPSAT, SWATH and SATPATH. A description of how to run each of the programs follows.

C.2.1 HOW TO RUN DRAWGLOBE

DRAWGLOBE must be run prior to the operation of the simulation model. This program creates the map projections that the simulation model will use later. The program has to be run only once as the map projections are stored in terminal memory to be recalled whenever they are needed. The maps will remain in terminal memory until the terminal is turned off or until RESET is entered in SETUP mode.

The following menu is displayed to determine which background map is to be drawn.

```
Enter choice of map.

Standard polar(P)                    Standard mercator(M)
Both standard maps(B)                Other(O)

>
```

The user has the flexibility of creating the standard polar map, mercator map or both. For these two options, no further input is required from the user. For the option, other(O), it is necessary for the user to input the map options in order to define the background map desired. The following message will be displayed.

```
INPUT MAP OPTIONS;  DEFAULT VALUES FOLLOW:
     TYP         1.000      VLAT        0.000
     VLON        0.000      LATMIN    -90.000
     LATMAX     90.000      LONMIN   -180.000
     LONMAX    180.000      DTASTP      5.000
     ISLLIM     20.000      BNDOP       0.000
     SCALE       1.000      GRID        0.000
     LATSTP     10.000      LONSTP     10.000
     SHORE       0.000
>
```

Input the parameters in the order indicated. (ie: typ, vlat, vlon, latmin,...)

C.2.2 HOW TO RUN SGPSAT

SGPSAT is the next program to run in the simulation model. This is the program that calculates the position of the satellites for a given simulation time. The user must input the epoch parameters as well as type(s) of output files to be created.

C.2.2.1 Epoch Parameters

The epoch parameters can be supplied to the SGPSAT program in two ways. One method is to create a data file of these parameters before running the SGPSAT program, while the other method is to enter these values interactively.

The input data file enables the user to rerun the simulation using the same data set if desired or altering the data set with minimal editing and typing. The name of the data file is EPOCHDATA.TXT. This text file format is illustrated in section C.3.

The epoch values are changed to the values desired before running the program. If there is more than one satellite in the simulation, repeat the list of parameters for next satellite exactly as the previous list.

If interactive input of the epoch parameters has been chosen, the program enters a screen editing mode. The following display appears.

INITIAL CONDITIONS

	DATE	TIME	JULIAN DATE
EPOCH	1 JAN 50	0 0.000	33282.000000
SIMULATION	1 JAN 50	0 0.000	33282.000000

TOTAL RUN(MINS) 0.000 DELTA TIME(MINS) 0.000

ORBITAL ELEMENTS

ARG OF PERIGEE	RT ASCENSION	INCLINATION
0.000000	0.000000	0.000000
MEAN ANOMOLY	ECCENTRICITY	MEAN MOTION
0.000000	0.000000	0.000000

TIME DERIVATIVES

1ST	0.000000
2ND	0.000000

The first time this screen appears, all parameters are initialized to '0.0'. These initial values cannot be used for the simulation. On subsequent runs, these parameters retain the values of the previous run. The keyboard's function keys are used to input the epoch data with the screen editor. Instructions on the use of the function keys appear as follows:

```
ENTER DATA IN REAL FORMAT
F1,F2,F3,F4 - SELECT DATA ITEM
F5 - SIGNIFIES DATA ENTRY
F8 - EXIT FROM EDITOR
```

To input data, the cursor is moved to the desired data item with the use of the cursor keys. Then, the F5 key is pressed to indicate to the program that the data value is to be altered. The new value is then entered as a real number. Press <RETURN> to indicate the termination of data entry and to cause the cursor to move to the next data item. F8 is

pressed to exit from the screen editor.

C.2.2.2 Running the SGPSAT Program

To invoke the program, enter the command,

$RUN SGPSAT

The following introductory screen and prompt appears.

```
*************************************************************************
*                                                                       *
*  SGPSAT:                                                              *
*  -------                                                              *
*                                                                       *
*      This program calculates a satellite flight path (elliptic)       *
*  based upon a set of epoch conditions.                                *
*                                                                       *
*      Output is either in the form of listings or a disc, or both.     *
*  Earth traces can be produced.                                        *
*                                                                       *
*      Listings:  Any of the following formats can be produced.         *
*                     a) Inertial cartesian coordinates (first point    *
*                        of aries)                                      *
*                     b) Earth cartesian (Greenwich)                    *
*                     c) Geographic, lat and long                       *
*                     d) Orbital coordinates                            *
*                                                                       *
*************************************************************************
IS THE DATA ON FILE ? (Y)=YES )
```

If a data file has been created a 'Y' is entered. This tells the program to receive the input parameters from the file. If the user has chosen not to use this file, a 'N' is issued. This tells the program to receive data from the keyboard.

Once the epoch parameters are input, the orbital parameters are calculated for the satellite. These parameters are displayed for the user as follows:

	DATE	TIME	RUN NO 1	
EPOCH	8 NOV 85	10 30.000	EPOCH DT (DAYS)	0.000
SIMULATION	8 NOV 85	10 30.000	TOTAL RUN TIME (MINS)	220.000
			DELTA TIME(MINS)	2.000

ORBITAL ELEMENTS

ARG OF PERIGEE	RT ASCENSION	INCLINATION
0.000000	0.000000	90.000000
SEMI MAJOR AXIS	ECCENTRICITY	MEAN MOTION
1.162613	0.000010	13.600000

INITIAL CONDITIONS

HOUR ANGLE	RX	RY	RZ
205.056	7416.720	0.000	-14.352
R	V	VR	VA
7416.733	7331.763	-7.092	7331.759

If the screen editing option has been chosen, the question

CHECK INPUT? (Y)=YES >

will appear. To return to the screen editor to make changes to the data set before the simulation is run, a 'Y' is entered. An entry of 'N' causes the program to proceed. Regardless of how the data is entered, the following prompt is displayed.

RUN THE SIMULATION? (Y)=YES >

To start the simulation, type 'Y'. The program issues the message

SIMULATION RUNNING

to indicate the commencement of the calculations. The following prompts are then displayed, one of which requires a response from the user.

CALCULATE APOGEE/PERIGEE & CROSSING TIMES? (Y)=YES >
PROGRAM IS BUSY

Next, the following prompts are displayed to determine which output files are to be created.

```
DO YOU WANT OUTPUT LISTINGS? (Y)=YES >
DO YOU WANT INERTIAL CARTESIAN? (Y)=YES >
DO YOU WANT EARTH CARTESIAN? (Y)=YES >
DO YOU WANT GEOGRAPHIC FORMAT? (Y)=YES >
DO YOU WANT ORBITAL FORMAT? (Y)=YES >
DO YOU WANT A RELATIVE TIME SCALE ON LISTINGS (Y)=YES >
```

Each of the first five questions must be answered by 'Y' in order for the rest of the simulation model to function properly. This action will cause the four types of output files to be created. The contents of these files are explained in Appendix D. If a relative time scale is desired, the following prompt appears:

```
OFFSET TIME(MINS), RELATIVE TO SIMULATION RUNNING TIME
>
```

and the user enters the desired offset time. The prompt

```
LIST FILE BEING MADE FOR THE SPOOLER
```

is displayed indicating that the data files are being created. Upon completion, the program gives the option of adding another satellite to the constellation. If the user has created a file of several satellites, this first run has only read the data for the first satellite. A response of 'Y' to the following prompt is required to process the next set of epoch data.

```
ANOTHER RUN? (Y)=YES >
```

The satellite constellation has been created. The next step in the simulation model is to calculate the radar coverage swaths.

C.2.3 HOW TO RUN SWATH

The third program of the simulation model is SWATH. This program calculates the radar coverage swaths for each of the satellites in the simulation. There is very little user interaction required to run this program.

The three input parameters required to calculate the radar coverage swaths are the minimum and maximum grazing angles for the radar, and the sector size of the radar's

circular 'footprint'. All of these values are input in degrees. To run this program enter the following command.

RUN SWATH

The following prompt is displayed:

Input the min, max grazing angles and sector size. >

The data is entered in the following format. Each of the numbers must have three digits before the decimal and two digits after, using zeros as place holders where needed. For example, a value of 30 is entered as 030.00 while 180 is entered as 180.00. The numbers must be separated by a space or comma.

Next, it is necessary to provide the input parameters for the airborne target. The following prompts are displayed.

Latitude and Longitude of Start Position >
Bearing Angle of Flight Path >
Altitude of Target(metres) >
Velocity of Target(km/h) >

The data are entered as real numbers with the latitude and longitude separated by a space or a comma. Two output files, Geocor.dat and Trgcor.dat are created to be used as input to the SATPATH program. A description of these files is provided in Appendix D.

C.2.4 HOW TO RUN SATPATH

The fourth program of the simulation package is SATPATH. This program generates the graphic output of the simulation data. The user is required to respond to several prompts, which will determine the style of graphical output.

This program has the capability to display ground traces on both the polar and mercator maps, airborne target detection on both the polar and mercator maps, and orbital traces on the polar map. To run this program, enter the command,

RUN SATPATH

The following menu is displayed. Enter the number corresponding to the task to be performed.

```
SELECT ONE:
    1) Polar orbit traces
    2) Polar ground traces
    3) Polar target detection
    4) Mercator ground traces
    5) Mercator target detection

SELECTION =
```

No further user interaction is required.

C.2.5 HOW TO RUN ERASE

The final program provided in the simulation package is ERASE. This package erases the graphics screen of the Tektronix 4107 terminal. To run this program, enter:

```
RUN ERASE
```

No further user interaction is required.

C.3 SAMPLE RUN OF THE SIMULATION PACKAGE

A sample run of the entire simulation is provided to illustrate the process of producing graphic displays of the simulation model. In this sample simulation, the background maps are drawn, a file of epoch parameters setup, and the three programs run in order to generate the ground trace of the simulation satellite on the polar map.

```
$ RUN DRAWGLOBE

 Enter choice of map.

 Standard polar(P)          Standard mercator(M)
 Both standard maps(B)      Other(0)

 >B
```

For this simulation, a file of parameters has been set up in EPOCHDATA.TXT as follows.

```
{------------------------------------------------------------------------}
{NOTE : 1) ALL Numeric Data Is In FLOATING POINT Format                   }
{       2) ALL Data MUST Terminate With A ','                             }
{                                                                         }
{*  FORMAT For Entering Julian Date - Enter Modified Julian Date          }
{   FORMAT For Entering Standard Date - Enter DAY MONTH YEAR ZULU_TIME    }
{                ZULU_TIME Has The Form - HHMM.MM                         }
{   (SAMPLE) 22 APR 65 2359.00,    Means Exactly 11:59 PM. On APRIL 22,1965 }
{                                                                         }
{NOTE : This Header MUST Remain 13 Lines Long                             }
{        And Leave NO Space Between The End Of One Series Of              }
{         Epoch Conditions And The Start Of The Next.                     }
{------------------------------------------------------------------------}
0.0,                          MEAN ANOMALY
13.60,                        MEAN MOTION
0.0,                          MEAN MOTION TIME DERIVATIVE 1
0.0,                          MEAN MOTION TIME DERIVATIVE 2
0.0,                          RIGHT ACSENSION
0.0,                          ARGUMENT OF PERIGEE
90.0,                         INCLINATION
0.00001,                      ECCENTRICITY
S,                            J=JULIAN  S=STANDARD TIME ON NEXT LINE
08 NOV 85 1030.00,            EPOCH DATE - FORMAT *
S,                            J=JULIAN  S=STANDARD TIME ON NEXT LINE
08 NOV 85 1030.00,            SIMULATION START TIME - FORMAT *
220.0,                        TOTAL RUN TIME - MIN
2.0,                          DELTA T - MIN
180.0,                        MEAN ANOMALY
13.60,                        MEAN MOTION
0.0,                          MEAN MOTION TIME DERIVATIVE 1
0.0,                          MEAN MOTION TIME DERIVATIVE 2
0.0,                          RIGHT ACSENSION
0.0,                          ARGUMENT OF PERIGEE
90.0,                         INCLINATION
0.00001,                      ECCENTRICITY
S,                            J=JULIAN  S=STANDARD TIME ON NEXT LINE
08 NOV 85 1030.00,            EPOCH DATE - FORMAT *
S,                            J=JULIAN  S=STANDARD TIME ON NEXT LINE
08 NOV 85 1030.00,            SIMULATION START TIME - FORMAT *
220.0,                        TOTAL RUN TIME - MIN
2.0,                          DELTA T - MIN
```

```
 SIMULATION RUNNING
  CALCULATE APOGEE/PERIGEE & CROSSING TIMES? (Y)=YES >N
 PROGRAM IS BUSY

  DO YOU WANT OUTPUT LISTINGS? (Y)=YES >Y
  DO YOU WANT INERTIAL CARTESIAN? (Y)=YES >Y
  DO YOU WANT EARTH CARTESIAN? (Y)=YES >Y
  DO YOU WANT GEOGRAPHIC FORMAT? (Y)=YES >Y
  DO YOU WANT ORBITAL FORMAT? (Y)=YES >Y
  DO YOU WANT A RELATIVE TIME SCALE ON LISTINGS (Y)=YES >N
 LIST FILE BEING MADE FOR THE SPOOLER
  ANOTHER RUN? (Y)=YES >N



$ RUN SWATH

 Input the min, max grazing angles and sector size.
        >005.00 030.00 090.00

 INPUT THE INITIAL PARAMETERS OF THE AIRBORNE TARGET


 Latitude and Longitude of Start Position > 20.0, -60.0
 Bearing Angle of Flight Path > 0.0
 Altitude of Target(metres) > 6096.0
 Velocity of Target(km/h) > 10000.0


   Creating the files...

FORTRAN STOP

$ RUN SATPATH


   SELECT ONE:
      1> Polar orbit traces
      2> Polar ground traces
      3> Polar target detection
      4> Mercator ground traces
      5> Mercator target detection

   SELECTION = 2


FORTRAN STOP
$
```

The output that is generated from this run is illustrated in Figure C1.

Figure C1: Sample Output

APPENDIX D

Data Dictionary of Arrays and Files

D.1 SGPDAT ARRAY

This array is used to store orbital parameters in the SGPSAT program. This one dimensional array of 14 elements is created in the SGP module. The elements of the array are:

1) Current Perigee
2) Current Rt Ascension
3) Current Inclination
4) Semi major Axis
5) Eccentricity
6) Mean Motion
7) Hour Angle
8) Rx
9) Ry
10) Rz
11) R
12) V
13) dR/dt
14) rdV/dt

Angles are given in terms of radians, length in units of earth radii, and velocity in units of earth radii/minute.

D.2 FD ARRAY

The FD array is generated in the SGPRUN module of the SGPSAT program. This array stores information from each computational cycle in order to produce the output files. This 3x7 matrix contains the following data.

ROW	Column 1	2	3	4	5	6	7
1	Epoch Time	Arg Perigee	Semi Axis	Hour Angle	Radial V	Rx	Vx
2	Start Time	Rt Ascension	Eccentricity	R(km)	Azimuth V	Ry	Vy
3	Running Time	Inclination	Mean Motion	V(m/sec)	-	Rz	Vz
UNITS		radians			m/sec	km	m/sec

Note that the position vector R(Rx,Ry,Rz) and the velocity vector V(Vx,Vy,Vz) are in Inertial reference frame coordinates. Given the Hour Angle, i.e. FD(1,4), this data can be converted to an Earth Cartesian system and hence into a Geographic system in the SGPFMT subroutine (found in the SGPSAT main routine). Similarly, the Euler angles (perigee, ascension, inclination) are used to convert the R and V vectors into an Orbital coordinate system in the routine ECOECI(found in the SGPSAT main routine).

D.3 FILEDATA, DATA ARRAYS AND GEOCOR.DAT FILE

The purpose of the 3 dimensional FILEDATA and DATA arrays is to store the latitude and longitude points of one circle for each of the satellites in the current simulation. Their format is similar to the GEOCOR.DAT file, which FILEDATA creates in SWATH and from which DATA is loaded in SATPATH. Figure D1 shows a portion of a GEOCOR.DAT file.

SATELLITE 1		SATELLITE 2		
LAT	LONG	LAT	LONG	
55.660	105.576	55.660	105.576	1
59.616	102.050	59.616	102.050	2
63.050	96.364	63.050	96.364	3
65.653	88.204	*65.653*	88.204	4
67.080	77.852	67.080	77.852	5
67.080	66.584	67.080	66.584	6
65.653	56.232	65.653	56.232	7
63.050	48.073	63.050	48.073	8
59.616	^42.386^	59.616	42.386	9
55.660	38.860	55.660	38.860	10
9999.000	9999.000	9999.000	9999.000	11
48.969	84.986	48.969	84.986	12
50.501	82.880	50.501	82.880	13
51.729	80.266	51.729	80.266	14
52.587	77.233	52.587	77.233	15
53.030	73.922	53.030	73.922	16
53.030	70.514	53.030	70.514	17
52.587	67.204	52.587	67.204	18
51.729	64.171	51.729	64.171	19
50.501	61.556	50.501	61.556	20
48.969	59.450	48.969	59.450	21

9999.000	9999.000	9999.000	9999.000	22
55.660	105.576	55.660	105.576	23
48.969	84.986	48.969	84.986	24
9999.000	9999.000	9999.000	9999.000	25
55.660	38.860	55.660	38.860	26
48.969	59.450	48.969	59.450	27
9999.000	9999.000	9999.000	9999.000	28
99999.000	99999.000	99999.000	99999.000	29

Figure D1: GEOCOR.DAT file

The first column contains the latitudes (in degrees) of the points on the coverage circle of the first satellite. The second column contains the longitudes (also in degrees) of the points of the coverage circle of the first satellite. Similarly, the second set of columns hold the latitudes and longitudes of points of the coverage circle of the second satellite. Any successive satellites (up to 15 in total) will produce additional pairs of columns of this same format.

The indexing of the arrays FILEDATA and DATA arrays is as follows. The first index in the array indicates the current point to plot on this circle (in Figure D1 this number ranges from 1 to 29). The second index of the array contains a code (1 or 2), where 1 indicates the latitude of the point on the circle, and 2 indicates the longitude of the point on the circle. The final index of the array specifies the current satellite number.

Thus the element in Figure D1 marked by a '*' will be FILEDATA(4,1,2) - the latitude of the fourth point of the coverage circle of satellite number 2

Similarly, FILEDATA(9,2,1) is marked by a '^' and it is the longitude of the ninth point of first satellite's coverage circle.

The 9999.000 indicates to the plotter (in SATPATH) to lift its pen, while the 99999.000 indicates to the SATPATH program that a coverage circle is complete.

D.4 TRGCOR.DAT FILE

The file, TRGCOR.DAT is created in the program SWATH. It contains the target path data that is used in SGPSAT to illustrate the target and to determine if it has been detected by the satellite. A sample file is provided in Figure D2.

```
1  20.00  300.00  0.00000  6.096  2999.61  -5195.48  2183.54
1  23.00  300.00  0.00000  6.096  2938.37  -5089.41  2494.47
1  26.00  300.00  0.00000  6.096  2869.08  -4969.40  2798.57
```

Figure D2: TRGCOR.DAT file

The first column contains a flag indicating the flight path. It can contain three values: 1 - flight path heading north on a meridian, 2 - flight path heading south on a meridian, and 3 - flight path not on a meridian. The second and third columns contain the latitude and longitude of the target respectively, the fourth column the bearing angle of the target, the fifth column the height in kilometres and the last three columns contain the X, Y and Z coordinates of the target in the earth rotating reference frame.

D.5 DETECTPOS.DAT FILE

The file, DETECTPOS.DAT is created in the program SATPATH. A sample output follows:

```
3  10.00  35.00  60.00  1037.00
```

The first column contains the satellite number, the second column contains the time of the target detection, the third and fourth columns contain the latitude and longitude of the target in degrees and the last column contains the height of the target in kilometers. All this data is recorded in the file upon detection of the target during the simulation run.

APPENDIX E

E.1 ESCAPE SEQUENCES

The following is a list of the Tektronix 4107 Colour Graphics terminal escape sequences that are used in the implementation of the orbital model.

Command	Escape Sequence
Move	Esc LF Xposition Yposition
Draw	Esc LG Xposition Yposition
Select Code	Esc %! Integer
Clear Dialog Area	Esc LZ
Enable Dialog Area	Esc KA Integer
Set Number of Dialog Lines	Esc LL Integer
Page	Esc Formfeed
Open Segment	Esc SO Integer
Select Pivot Point	Esc SP Xposition Yposition
Close Segment	Esc SC
Select View	Esc RC Integer
Set Surface Definitions	Esc RD Integer
Set Segment Visibility	Esc SV Segment Number, Visibility Code
Set View Attributes	Esc RA Surface Number, Wipe Index, Border Index
Set Dialog Visibility	Esc LV Integer
Set Surface Visibility	Esc RI2 Surface Number, Enable Mode
Renew View	Esc KN Integer
Line Index	Esc ML Integer
Cursor Position	Esc [Row number ; Column number H
Edit Margins	Esc [Top margin ; Bottom margin r
Graphic Rendition	Esc [Integer m
Erase in Display	Esc [Erase mode J
Reset Origin Mode	Esc [?61
Set Queue Size	Esc [integer] NQ

APPENDIX F

Program Listings

```
$!-----------------------------------------------
$!
$! Procedure  M A P L I N K
$!
$!   Date:           Author:
$!
$! September 1985   Mark Lovell
$!
$!-----------------------------------------------
$!
$!  Description:
$!      This procedure links all of the modules
$!      needed to run the DRAWGLOBE program
$!
$!   Linking Sequence:
$!       @maplink
$!
$!-----------------------------------------------
$
$    !MAPLINK
$
$Link drawglobe,graphics,graphics1,[plotter]plot10/lib
$
$    !CEND of MAPLINK]
```

```
C*************************************************************************
      PROGRAM MAPTEST
C*************************************************************************
      CHARACTER*6 NAMES(15)
      REAL LATMIN,LATMAX,LONMIN,LONMAX,LATSTP,LONSTP
      REAL XLAT,XLON,ZLEN,XPLOT,YPLOT,ZPLOT,VLAT,VLON
      REAL FLAT,FLON,FALT,DARX(4),DARY(4),UARX(4),UARY(4)
      INTEGER ERROR,SIZE
      INTEGER DTASTP,OP,BNDOP,SHORE,GRID,STOP,OUT,IN,SEGNO,VIS
      CHARACTER*1 CHOICE,ANS
      DIMENSION  STORE(15),XP(500),YP(500),XX(31),YY(31)
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
      COMMON/S/STOP
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      DATA NAMES/'TYP','VLAT','VLON','LATMIN','LATMAX',
     3           'LONMIN',
     1     'LONMAX','DTASTP','ISLLIM','BNDOP','SCALE',
     2     'GRID','LATSTP','LONSTP','SHORE'/
      DATA STORE / 1. ,0. ,0. ,-90. ,90. ,-180. ,180. ,5. ,20 .
     1 ,0. ,1. ,0. ,10. ,10. ,0. /

      OUT = 6
      IN  = 5
      SIZE = 4000
C
C     Change the terminal to tek mode
C
      CALL SELECT_CODE(0)
      CALL CLEAR_DIALOG_AREA
C
C     Change input queue size
      CALL QUEUESIZE(SIZE)
C
      WRITE(OUT,310)
 310  FORMAT(///,' Enter choice of map. ',//,' Standard polar(P) ',19X,
     1 'Standard mercator(M) ',/,' Both standard maps(B) ',15X,
     1 'Other(O) ',//,'$) ')
      READ(IN,320)CHOICE
 320  FORMAT(A1)
C
 331  CONTINUE
C
      CALL CLEAR_DIALOG_AREA
C
 332  CONTINUE
C
      CALL INFIN(9998.)
      STOP=0
C
      IF ((CHOICE .EQ.'O') .OR.(CHOICE .EQ. 'o')) THEN
              WRITE(OUT,9999)
9999                  FORMAT(' INPUT MAP OPTIONS:  DEFAULT VALUES FOLLOW:')
                      WRITE(OUT,9998)(NAMES(I),STORE(I),I=1,15)
9998                  FORMAT(2(5X,A6,F10.3))
                      READ(IN,*)STORE
                      VLAT = STORE(2)
                      VLON = STORE(3)
                      LATMIN = STORE(4)
                      LATMAX =  STORE(5)
                      LONMIN = STORE(6)
                      LONMAX =  STORE(7)
              IF( STORE( 1 ) .EQ. 1.)THEN
                      TYP   = 4HORT
                      SEGNO = 1
                      ELSE IF( STORE( 1 ) .EQ. 2.)THEN
                      TYP   = 4HMCR
                      SEGNO = 2
                      ELSE
                      WRITE(OUT,340)
 340                    FORMAT(/,' The value entered as "TYP" must be ',
     1                     'either a 1 (orthographic) ',/,' or a ',
     1                   '2 (mercator). ',///)
                      GO TO 332
```

```
                    END IF
                    DTASTP = INT(STORE(8))
                    ISLLIM = INT(STORE(9))
                    BNDOP  = INT(STORE(10))
                    SCALE  = STORE(11)
                    GRID   = INT(STORE(12))
                    LATSTP = STORE(13)
                    LONSTP = STORE(14)
            SHORE  = INT(STORE(15))
      ELSE
            VLON   = 0.0
            LONMIN = -180.0
            LONMAX = 180.0
            DTASP  = 5
            ISLIM  = 40
            BNDOP  = 0
            SCALE  = 2
            GRID   = -1
            LATSTP = 10.0
            LONSTP = 10.0
            SHORE  = 1
C
            IF ((CHOICE .EQ. 'P') .OR. (CHOICE .EQ. 'p') .OR.
     1                  (CHOICE .EQ. 'B') .OR. (CHOICE .EQ. 'b')) THEN
                    SEGNO  = 1
                    TYP    = 4HORT
                    VLAT   = 90.0
                    LATMIN = -90.0
                    LATMAX = 90.0
                    ELSE IF ((CHOICE .EQ. 'M') .OR. (CHOICE .EQ. 'm')) THEN
                    SEGNO  = 2
                    TYP    = 4HMCR
                    VLAT   = 0.0
                    LATMIN = -85.0
                    LATMAX = 85.0
                    ELSE
                    WRITE(OUT,350)CHOICE
  350               FORMAT(/,' The value ',A1,' entered as map choice',
     1                       ' is invalid.')
                    STOP
            END IF
      END IF
C
C
C     set surface definitions
      CALL SDEFN(1,1,1,1,ERROR)
      IF(ERROR) STOP
C
C     select view one
      CALL VSELECT(1,ERROR)
      IF(ERROR) STOP
C
C     set view attributes
      CALL VATTRIB(1,0,1,ERROR)
      IF (ERROR) STOP
C
C     define a segment
      call openseg(SEGNO)
C
      PRINT *,' ***PLOTTING***'
      CALL MAP(TYP,VLAT,VLON,LATMIN,LATMAX,LONMIN,LONMAX,
     1 DTASTP,ISLLIM,BNDOP,SCALE,GRID,LATSTP,LONSTP,
     1 SHORE)
C
C     Close the defined segment
C
      CALL SGCLOSE
      CALL PLOTEND
C
C     Turn on the dialog area
C
      CALL DAVIS(1,ERROR)
      IF (ERROR) STOP
C
      WRITE(OUT,375)
```

```
 375  FORMAT(/,'  Map is defined on a segment.')
C
C     Turn off the segment visibility
      VIS = 0
      CALL SGVIS(SEGNO,VIS,ERROR)
C
      IF ((CHOICE .EQ. 'B') .OR. (CHOICE .EQ. 'b')) THEN
                     CHOICE = 'M'
                     GO TO 331
      END IF
C
C     Change queue size back
      SIZE = 300
      CALL QUEUESIZE(SIZE)
C
C     Change back to ANSII mode
C
      CALL SELECT_CODE(1)
C
      STOP
      END
C
C LAST UPDATE 11 JUNE 1980 BY .DAB. MAYBE COMPLETE .
C THIS PLOTS A UNIT SPHERE,NO MULTIPLES OF 3 REQUIRED .
C MAPS (WITHOUT ANY USER ADDED POINTS OR LINES) MAY BE PLOTTED
C USING THESE ROUTINES AND PROGRAM MAPTEST ON  DAB.USL
C A COMPILED... VERSION OF MAPTEST IS ON DAB.TM
C*************************************************************************
C*************************************************************************
C THESE SUBROUTINES PLOT THE WORLD MAP.
C*************************************************************************
C*************************************************************************
C
C
C
C*************************************************************************
      SUBROUTINE MAP(G,A,B,C,D,E,F,I,K,L,O,Q,R,S,U)
C*************************************************************************
C THIS SUBROUTINE SETS ALL CONSTANTS THAT AREN'T PROPERLY SET AND
C CALLS THE SUBROUTINES TO :
C                    1. SETFLAGS
C                    2. CONVERT DEGREES TO RADIANS
C                    3. SET THE VIEWPOINT
C                    4. SET THE PROJECTION MATRIX
C                    5. SET UP TEKTRONIX GRAPHICS PARAMETERS
C                    6. DRAW OUTLINES
C                    7. PLOT MAP
C                    8. DUMP FINAL PART OF PLOT IN BUFFER
C                       TO THE SCREEN .
C
      DIMENSION FILE107(5),FILE700(5)
      REAL LATMIN,LATMAX,LONMIN,LONMAX,LATSTP,LONSTP,MODE
      INTEGER DTASTP,BNDOP,OP,Q,GRID,SHORE,U
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      DATA PI/ 3.141592654 /
C
C OPTIONS SET HERE IF NOT PROPERLY ENTERED.
C
C   DTASTP
      IF(I.LT.1)I=1
C   OP
      J=0
      IF(A .EQ. 90.)J = 1
      IF(A .EQ. -90. )J = -1
      IF(J.EQ.1)D=90.
      IF(J.EQ.-1)C=-90.
C   BNDOP
      IF(L.NE.1)L=0
C   ISLLIM
      IF(K.LT.2)K=2
C   SHORE
      IF(U.NE.0)U=1
```

```
C
C CHOSE TYPE OF ORT IF ORT IS CHOSEN .
      ITEST = 0
      IF( (D-C) .EQ. 180. .AND. (F-E) .EQ. 360. )ITEST = 1
      IF( ITEST .EQ. 1 .AND. G .EQ. 4HORT )G = 4HORTT
      IF( ITEST .EQ. 0 .AND. G .EQ. 4HORT )G = 4HORTP
C
C INPUT DATA ASSIGNED TO THE COMMON BLOCK HERE.
C
      VLAT=A
      VLON=B
      LATMIN=C
      LATMAX=D
      LONMIN=E
      LONMAX=F
      TYP=G
      DTASTP=I
      OP=J
      ISLLIM=K
      BNDOP=L
      SCALE=O
      GRID=Q
      LATSTP=R
      LONSTP=S
      SHORE=U
C
C SET UP TAPE IF 'SHORE' OPTION SELECTED .
C
      IF (SHORE .EQ. 1) OPEN(107,NAME='[SLINN.GLOBE]MAP2.DAT',
     1 ACCESS='SEQUENTIAL',IOSTAT=IOS,
     2 STATUS='OLD',RECL=10000)
      IF( SHORE .EQ. 1 )REWIND 107
C
       CALL SETFLAGS
C
      CALL RADIANCONVERSION
C
      CALL VIEWPOINT
C
      CALL MATRIXSET
C
      CALL PLOTSETUP
C
      CALL MAPPLOT(ISLLIM,BNDOP,SHORE)
C
      CALL PLOTEND
C
      CLOSE(107)
C      CLOSE(700)
      RETURN
      END
C
C*************************************************************************
      SUBROUTINE MAPPLOT(ISLLIM,BNDOP,SHORE)
C*************************************************************************
C THIS SUBROUTINE DOES THE ACTUAL DRAWING OF THE MAPS.
C
      DIMENSION POLE(3),XXXX(2),YYYY(2)
      REAL LATMIN,LATMAX,LONMIN,LONMAX,LATSTP,LONSTP
      INTEGER ERROR,DTASTP,BNDOP,GRID,COUNT,OP,SHORE,PTSINCREMENTED
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/FLAGS/IFLAG1,IFLAG2,IFLAG3,IFLAG4
      COMMON/S/STO,
C
      DATA UNITLENGTH/1./
C
C  'NTEST' LATER CHECKED TO SEE IF LAST LINE NUMBER .
      NTEST = 0
C  'COUNT' INCREMENTS WHEN THE SHORE HAS BEEN PLOTTED .
      COUNT = 0
C
C MAP SETUP ROUTINES EXECUTED HERE:OUTLINES PLOTTED.
```

```
C
      CALL MAPSET( TYP )
C
C SHORE OPTION CHECKED HERE.
C
      IF(SHORE.EQ.0)GO TO 60
C
C  INITIALIZE NUMBER OF LINES PLOTTED .
C
      LINESEGMENT = 0
C
C THE STRING LENGTH AND OCCURANCE OF ISLAND ARE DETERMINED HERE.
C
    3 CALL GETVAL(ERROR,XX,YY,1)
      IF (ERROR .GT. 2) WRITE(6,99)ERROR
 99   FORMAT(' BEFORE NTESTS ',I5)
      IF (STOP .EQ.1)GO TO 300
   20 N1=XX+.5
      IF( NTEST .EQ. 700083 ) GO TO 300
      NTEST = YY+.5
C  'N' IS NUMBER OF POINTS IN THE NEXT LINE .
      N=MOD(N1,10000)
C  'N2' = 3  DESIGNATES AN ISLAND .
      N2=(N1-N)/10000
C
C ISLAND SIZE OPTION CHECKED HERE.
C
      IF(N2.EQ.3.AND.N.GE.ISLLIM)GO TO 30
      IF(N2.NE.3.AND.N.GE.2)GO TO 30
C
C INCREMENT LINE W/OUT PLOTTING IF LINE UNACCEPTABLE .
C
      PTSINCREMENTED = N + 1
      CALL GETVAL(ERROR,XX,YY,PTSINCREMENTED)
      IF (ERROR .NE. 2) WRITE(6,98)ERROR
 98   FORMAT(' 1',I5)
      GO TO 20
C
C CHECKS TO SEE IF THE NUMBER OF POINTS EXCEEDS THE SUBSCRIPT DIMENSION.
C
   30 IF(N.LE.500)GO TO 5
      WRITE(2,10)
   10 FORMAT(2X,'MORE THAN 500 POINTS')
      STOP
C
    5 CONTINUE
C
C  'N1' IS THE NUMBER OF THE POINT BEING CONSIDERED IN PRESENT LINE.
C
      N1=0
C
C LINESEGMENT HERE IS THE NUMBER OF THE LINE SEGMENT TO BE PLOTTED.
C
      LINESEGMENT = LINESEGMENT + 1
      IF(LINESEGMENT .EQ. -1)GO TO 300
C
      CALL GETVAL(ERROR,XX,YY,1)
       IF (ERROR .NE. 2) WRITE(6,97)ERROR
 97   FORMAT(' 2',I5)
C
C NEXT STATEMENT ASSIGNS MAP POSITION OR ERROR CONSTANT
C FOR THE FIRST POINT IN A STRING.
C
      CALL POINTPLOT(4HANG ,XX,YY,UNITLENGTH,XDUM,YDUM,ZDUM)
      IF (STOP .EQ. 1)GO TO 300
C
      J=1
C
C DATA OBTAINED USING DTASTP TO SKIP DATA POINTS.
C
   40 IF((J+DTASTP).GE.N)GO TO 50
      J=J+DTASTP
      CALL GETVAL(ERROR,XX,YY,DTASTP)
      IF (ERROR .NE. 2) WRITE(6,96)ERROR
 96   FORMAT(' 3',I5)
```

```
C
C NEXT STATEMENT ASSIGNS MAP POSITION OR ERROR CONSTANT
C FOR ALL BUT THE FIRST AND LAST POINTS IN A STRING.
C
      CALL POINTPLOT(4HANG ,XX,YY,UNITLENGTH,XDUM,YDUM,ZDUM)
      IF (STOP .EQ. 1) GO TO 300
C
      GO TO 40
C
   50 CONTINUE
C
C THE NEXT 2 LINES FIND THE ENDPOINT.
      ITR=N-J
      CALL GETVAL(ERROR,XX,YY,ITR)
      IF (ERROR .NE. 2) WRITE(6,95)ERROR
 95   FORMAT(' 4',I5)
C
C NEXT STATEMENT ASSIGNS MAP POSITION OR ERROR CONSTANT
C FOR THE LAST POINT IN A STRING.
C
      CALL POINTPLOT(4HANG ,XX,YY,UNITLENGTH,XDUM,YDUM,ZDUM)
      IF (STOP .EQ. 1) GO TO 300
C
C  OUTPUT PRESENT LINE .
C
      CALL LINE(0)
      IF( N1 .GT. 1 )CALL NPTS(N1)
      IF( N1 .GT. 1 )CALL CPLOT(X,Y)
C
      IF (STOP .EQ. 1)GO TO 300
      GO TO 3
  300 COUNT=COUNT+1
      IF(COUNT.EQ.1.AND.BNDOP.EQ.1)GO TO 3
   60 IF(GRID.NE.-1)CALL MESH
      CONTINUE
C
      RETURN
      END
C
C*************************************************************************
      SUBROUTINE ORTSET
C*************************************************************************
C THIS SUBROUTINE PLOTS THE OUTLINE(CIRCLE) OF A TOTAL GLOBE
C ORTHOGRAPHIC MAP BY CONNECTING 'IN1' POINTS AROUND THE CIRCLE.
C
      REAL LATMAX,LATMIN,LONMAX,LONMIN,LATSTP,LONSTP
      INTEGER OP,GRID,DTASTP
      DIMENSION POLE(3)
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
      DATA PI /3.141592654/
C
C IN1 IS # OF POINTS ON THE CIRCLE .
C
C
      IN1 = 100
      DO 1 K=1,IN1
      ANG = K / ( .5 * ( IN1 - 1 ) ) * PI
      X(K)= COS(ANG)
    1 Y(K)= SIN(ANG)
C
C THIS PLOTS A STRING X(I),Y(I) OF IN1 VARIABLES.
C
      CALL LINE(0)
      IF( IN1 .GT. 1 )CALL NPTS(IN1)
      IF( IN1 .GT. 1 )CALL CPLOT(X,Y)
C
      RETURN
      END
C
C*************************************************************************
      SUBROUTINE GETVAL(ERROR,X,Y,STEP)
```

```
C*********************************************************************
C
C THIS RETURNS A LATITUDE-LONGITUDE DATA PAIR WHEN CALLED , AND READS
C IN NEW BLOCKS OF DATA FROM THE TAPE WHEN THE PRESENT BLOCK HAS BEEN
C COMPLETELY READ.
C     DIMENSION A(2500)
      CHARACTER*4 A(2500)
      INTEGER ERROR,STEP
      COMMON/FLAGS/IFLAG1,IFLAG2,IFLAG3,IFLAG4
      IFLAG4=IFLAG4+STEP
    2 IF(IFLAG4.GT.500)GO TO 3
      JF=1+5*(IFLAG4-1)
      DECODE(20,1000,A(JF))X,Y
1000  FORMAT(2F10.3)
      RETURN
    3 IFLAG4=IFLAG4-500
C*****      CALL BUFFERIN(107,0,A,2500,ERROR,NWORDS,I)
      READ(107,1001)NWORDS,(A(II),II=1,2500)
1001  FORMAT(Q,2500A4)
      I = 0
      ERROR=2
      IF (ERROR .NE. 2)PRINT ERROR,I
      IF (I .NE. 0) PRINT ERROR,I
      GO TO 2
C
      END
C
C*********************************************************************
      SUBROUTINE MCRSET
C*********************************************************************
C THIS DRAWS THE OUTLINE(BOX) FOR THE MERCATOR AND CYLINDRICAL EQUAL
C AREA MAPS .
      REAL SCALE
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,DP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
C COMPUTES GRAPH OUTLINE DIMENSIONS.
C
      A = 4. / 3.
      B = 1.
C
C DRAWS GRAPH OUTLINE.
C
      X(1)=-A
      X(2)=A
      X(3)=A
      X(4)=-A
      X(5)=-A
C
      Y(1)=-B
      Y(2)=-B
      Y(3)=B
      Y(4)=B
      Y(5)=-B
C
      CALL LINE(0)
      CALL NPTS(5)
      CALL CPLOT(X,Y)
C
      RETURN
      END
C
C
C*********************************************************************
      SUBROUTINE SPRSET
C*********************************************************************
C THIS SUBROUTINE PLOTS THE OUTLINE FOR THE GNOMONIC,STEREOGRAPHIC,
C AZIMUTHAL EQUIDISTANT,& PARTIAL GLOBE ORTHOGRAPHIC MAPS .
C IF A POLAR VIEW WAS SELECTED THEN ONLY 'LATMIN' OR 'LATMAX' IS
C USED TO DETERMINE THE OUTLINE(WHICH ONE DEPENDS ON WHICH POLE) ;
C IF A NONPOLAR VIEW WAS SELECTED THEN 'LATMIN','LATMAX','LONMIN', &
C 'LONMAX' DETERMINE THE OUTLINE.
      REAL LATMAX,LATMIN,LONMAX,LONMIN
```

```
      INTEGER OP
      DIMENSION PT(2,200)
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/DIM/ISET
C
      DATA PI,UNITLENGTH/ 3.141592654,1. /
C
      ISET = 0
C
      DIFLON = LONMAX - LONMIN
      IF(OP.NE. 0)GO TO 60
      IF( OP .EQ. 0 .AND. DIFLON .GT. 6.28 ) GO TO 60
      GO TO 80
C
C VLON IS SET AT ONE-HALF THE SUM OF LONMAX AND LONMIN
C
C FEEDS DATA INTO ARRAY 'PT' TO BE USED TO DRAW THE OUTLINE
C
   60 IF(OP.EQ.-1)ANG=LATMAX
      IF(OP.EQ.1)ANG=LATMIN
      IF( OP .EQ. 0 .AND. VLAT .LE. 0.0 )ANG = LATMAX
      IF( OP .EQ. 0 .AND. VLAT .GT. 0.0 )ANG = LATMIN
      DO 70 I=1,200
      PT(1,I)=ANG
      PT(2,I)=-PI+2.*PI*(I-1.)/199.
   70 CONTINUE
      GO TO 125
   80 DO 90 I=1,50
      K=I
      PT(1,I)=LATMAX
      PT(2,I)=LONMIN+(K-1)/49.*(LONMAX-LONMIN)
   90 CONTINUE
      DO 100 I=51,100
      K=I-50
      PT(1,I)=LATMAX-(K-1.)/49.*(LATMAX-LATMIN)
      PT(2,I)=LONMAX
  100 CONTINUE
      DO 110 I=101,150
      K=I-100
      PT(1,I)=LATMIN
      PT(2,I)=LONMAX-(K-1.)/49.*(LONMAX-LONMIN)
  110 CONTINUE
      DO 120 I=151,200
      K=I-150
      PT(1,I)=LATMIN+(K-1.)/49.*(LATMAX-LATMIN)
      PT(2,I)=LONMIN
  120 CONTINUE
C
C THIS DETERMINES THE HORIZONTAL AND VERTICAL BOUNDS OF THE DATA
C AND RESCALES THE POINTS ACCORDINGLY.THE SCALE FACTORS ARE USED
C LATER FOR PLOTTING GLOBE DATA.
C
  125 HORMIN=4.
      HORMAX=-4.
      VRTMAX=-4.
      VRTMIN=4.
      N1 = 0
      ISET = 0
      DO 130 I=1,200
      CALL POINTPLOT(4HANG ,PT(1,I),PT(2,I),UNITLENGTH,X(I),Y(I),DUM)
  130 CONTINUE
      ISET = 1
C
      DO 140 I = 1 , N1
      IF( X( I ) .EQ. 9999. )GO TO 139
      IF(X(I).LT.HORMIN)HORMIN=X(I)
      IF(X(I).GT.HORMAX)HORMAX=X(I)
      IF(Y(I).LT.VRTMIN)VRTMIN=Y(I)
      IF(Y(I).GT.VRTMAX)VRTMAX=Y(I)
  139 CONTINUE
  140 CONTINUE
C
```

```
      DO 150 I = 1 , N1
      IF( X(I) .EQ. 9999. )GO TO 149
      X(I)=(-1.+((X(I)-HORMIN)/(HORMAX-HORMIN))*2.)
      Y(I)=(-1.+((Y(I)-VRTMIN)/(VRTMAX-VRTMIN))*2.)
      H=HORMAX-HORMIN
      V1=VRTMAX-VRTMIN
      SCA=H/V1
      IF(H.GE.V1)Y(I)=Y(I)/SCA
      IF(H.LT.V1)X(I)=X(I)*SCA
  149 CONTINUE
  150 CONTINUE
C
      CALL LINE(0)
      IF( N1 .GT. 1 )CALL NPTS(N1)
      IF( N1 .GT. 1 )CALL CPLOT(X,Y)
C
C OUTLINE PLOTTED
C
      RETURN
      END
C
C*************************************************************************
      SUBROUTINE XYZPOLAR(MODE,XIN,YIN,ZIN,PLAT,PLON,PLENGTH)
C*************************************************************************
C THIS TRANSFORMS FROM X,Y,Z COORDINATES TO LATITUDE,LONGITUDE,ALTITUDE
C COORDINATES IF MODE = 4HCOR   : THE REVERSE IF MODE = 4HANG    .
C
      REAL MAG , MODE
C
C
      COMMON/S/STOP
      DATA PI/3.141592654/
C
      IF( MODE .EQ. 4HANG )GO TO 200
      IF( MODE .EQ. 4HCOR )GO TO 100
      STOP
  100 CONTINUE
C
      TXIN = XIN
      TYIN = YIN
      TZIN = ZIN
C
C THIS COMPUTES LATITUDE,LONGITUDE,MAGNITUDE OF ( XIN , YIN , ZIN )
C
      PLENGTH=SQRT(TXIN**2+TYIN**2+TZIN**2)
      MAG=SQRT(TXIN**2+TYIN**2)
C
C  POINT LATITUDE COMPUTED NEXT
C
      IF(MAG.EQ.0.0 .AND. TZIN.GT.0.0)PLAT=PI/2.0
      IF(MAG.EQ.0.0 .AND. TZIN.LT.0.0)PLAT=-PI/2.0
      IF(MAG.NE.0.0)PLAT=ATAN2(TZIN,MAG)
C
C  POINT LONGITUDE COMPUTED NEXT
C
      IF(TXIN.EQ.0.0.AND.TYIN.EQ.0.0)PLON=0.0
      IF(TXIN.EQ.0.0.AND.TYIN.GT.0.0)PLON=PI/2.0
      IF(TXIN.EQ.0.0.AND.TYIN.LT.0.0)PLON=-PI/2.0
      IF(TXIN.NE.0.0)PLON=ATAN2(TYIN,TXIN)
C
      RETURN
C
  200 CONTINUE
C THIS FINDS THE ( X,Y,Z ) COORDINATES OF THE POINT WITH
C PLAT  LATITUDE AND PLON  LONGITUDE ON A PLENGTH RADIUS SPHERE
C
      TPLAT = PLAT
      TPLON = PLON
      TPLENGTH = PLENGTH
      IF (TPLAT .GT. 4096)XIN=0
      IF (TPLAT .GT. 4096)YIN=0
      IF (TPLAT .GT. 4096)ZIN=0
      IF (TPLAT .GT. 4096) STOP=1
      IF (TPLAT .GT. 4096)RETURN
C
```

```
      XIN=COS(TPLAT)*COS(TPLON) * TPLENGTH
      YIN=COS(TPLAT)*SIN(TPLON) * TPLENGTH
      ZIN=SIN(TPLAT) * TPLENGTH
C
      RETURN
      END
C
C*************************************************************************
      SUBROUTINE MESH
C*************************************************************************
C
C THIS GENERATES AND PLOTS LATITUDE AND LONGITUDE LINES FOR EACH
C MAP TYPE.EACH SET OF LINES STARTS AT ZERO DEGREES AND STEPS IN
C LATITUDE OR LONGITUDE BY STEPS LATSTP OR LONSTP.
C
      REAL LAT,LON,LATSTP,LONSTP,TYP,MODE
     1 ,LONMAX,LONMIN,LATMAX,LATMIN
      INTEGER OP,DTASTP,GRID
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
      DATA PI,UNITLENGTH/3.141592654,1./
C
C  (GRID) IS THE TYPE OF LINE TO BE DRAWN.
C
      CALL LINE(GRID)
C
C IN1 IS THE NUMBER OF POINTS PER LONGITUDE LINE.
C
      IN1=200
C
C N2 IS THE NUMBER OF POINTS PER LATITUDE LINE.
C
      N2=100
C
      IN11=IN1-1
      N21=N2-1
C
C THIS SECTION PLOTS THE UPPER HEMISPHERE LONGITUDINAL LINES
C INCLUDING THE EQUATOR.EACH LINE CONTAINS 100 POINTS.
      LAT=-LATSTP
    4 LAT=LAT+LATSTP
      HAFPI=PI/2.
      IF(LAT.GT.HAFPI)GO TO 5
      DO 1 I=1,IN1
      IF(TYP.EQ.4HORTT)LON=(2.*(I-1)/FLOAT(IN11)-1.)*PI
      IF(TYP.NE.4HORTT)LON=LONMIN+(LONMAX-LONMIN)*(I-1.)/FLOAT(IN11)
      X(I)=LAT
      Y(I)=LON
    1 CONTINUE
      N1 = 0
      DO 2 I=1,IN1
C
C MAP TYPE COORDINATES SELECTED HERE.
C
      XT = X( I )
      YT = Y( I )
C
      CALL POINTPLOT(4HANG ,XT,YT,UNITLENGTH,X(I),Y(I),DUM)
C
    2 CONTINUE
C
C PLOT STARTS HERE.
C
      IF( N1 .GT. 1 )CALL NPTS(N1)
      IF( N1 .GT. 1 )CALL CPLOT(X,Y)
      GO TO 4
    5 CONTINUE
C
C THIS SECTION PLOTS THE LOWER HEMISPHERE LONGITUDINAL LINES
C NOT INCLUDING THE EQUATOR.EACH LINE CONTAINS 100 POINTS.
C
      LAT=0.0
```

```
   14 LAT=LAT-LATSTP
      HAFPIN=-PI/2.
      IF(LAT.LT.HAFPIN)GO TO 15
      DO 11 I=1,IN1
      IF(TYP.EQ.4HORTT)LON=(2.*(I-1)/FLOAT(IN11)-1.)*PI
      IF(TYP.NE.4HORTT)LON=LONMIN+(LONMAX-LONMIN)*(I-1.)/FLOAT(IN11)
      X(I)=LAT
      Y(I)=LON
   11 CONTINUE
      N1 = 0
      DO 12 I=1,IN1
C
C MAP TYPE COORDINATES SELECTED HERE.
C
      XT = X(I)
      YT = Y(I)
      CALL POINTPLOT(4HANG ,XT,YT,UNITLENGTH,X(I),Y(I),DUM)
C
   12 CONTINUE
C
C PLOT STARTS HERE.
C
      IF( N1 .GT. 1 )CALL NPTS(N1)
      IF( N1 .GT. 1 )CALL CPLOT(X,Y)
      GO TO 14
   15 CONTINUE
C
C THIS SECTION PLOTS THE RIGHT HEMISPHERE LATITUDINAL LINES.
C EACH LINE CONTAINS 100 POINTS.
C
      LON=-LONSTP
   24 LON=LON+LONSTP
      IF(LON.GT.PI)GO TO 25
      DO 21 I=1,N2
      IF(TYP.EQ.4HORTT)LAT=((I-1)/FLOAT(N21)-.5)*PI
      IF(TYP.NE.4HORTT)LAT=LATMIN+(LATMAX-LATMIN)*(I-1.)/FLOAT(N21)
      X(I)=LAT
      Y(I)=LON
   21 CONTINUE
      N1 = 0
      DO 22 I=1,N2
C
C MAP TYPE COORDINATES SELECTED HERE.
C
      XT = X(I)
      YT = Y(I)
      CALL POINTPLOT(4HANG ,XT,YT,UNITLENGTH,X(I),Y(I),DUM)
C
   22 CONTINUE
C
C PLOT STARTS HERE.
C
      IF( N1 .GT. 1 )CALL NPTS(N1)
      IF( N1 .GT. 1 )CALL CPLOT(X,Y)
      GO TO 24
   25 CONTINUE
C
C THIS SECTION PLOTS THE LEFT HEMISPHERE LATITUDINAL LINES.
C EACH LINE CONTAINS 100 POINTS.
C
      LON=0.0
   34 LON=LON-LONSTP
      PIN=-PI
      IF(LON.LT.PIN)GO TO 35
      DO 31 I=1,N2
      IF(TYP.EQ.4HORTT)LAT=((I-1)/FLOAT(N21)-.5)*PI
      IF(TYP.NE.4HORTT)LAT=LATMIN+(LATMAX-LATMIN)*(I-1.)/FLOAT(N21)
      X(I)=LAT
      Y(I)=LON
   31 CONTINUE
      N1 = 0
      DO 32 I=1,N2
C
C MAP TYPE COORDINATES SELECTED HERE.
C
```

```
      XT = X(I)
      YT = Y(I)
      CALL POINTPLOT(4HANG ,XT,YT,UNITLENGTH,X(I),Y(I),DUM)
C
   32 CONTINUE
C
C PLOT STARTS HERE.
C
      IF( N1 .GT. 1 )CALL NPTS(N1)
      IF( N1 .GT. 1 )CALL CPLOT(X,Y)
      GO TO 34
   35 CONTINUE
C
      RETURN
      END
C*********************************************************************
      SUBROUTINE MAPSET( TYP )
C*********************************************************************
C THIS SUBROUTINE CALLS THE PROPER SUBROUTINE TO PLOT THE OUTLINE
C FOR A MAP OF TYPE 'TYP' .
      IF(TYP.EQ.4HORTT)CALL ORTSET
      IF(TYP.EQ.4HMCR .OR. TYP.EQ.4HCEQA)CALL MCRSET
      IF(TYP.EQ.4HGNM .OR.TYP.EQ.4HSTR .OR.TYP.EQ.4HAZEQ
     1 .OR.TYP.EQ.4HORTP)
     1 CALL SPRSET
      RETURN
      END
C
C*********************************************************************
      SUBROUTINE POINTPLOT(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C*********************************************************************
C THIS SUBROUTINE SUCCESSIVELY EXECUTES A '___PLOT' SUBROUTINE
C AND A '___CHECKANDSET' ROUTINE OF THE TYPE USED TO DRAW THE MAP
C IN SUBROUTINE MAP .
C
      REAL MODE,LATMAX,LATMIN,LONMAX,LONMIN,LATSTP,LONSTP
      INTEGER OP,DTASTP,GRID
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
      CALL PTCOR(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C
      CALL PTCHECK(XPLOT,YPLOT,ZPLOT)
C
      RETURN
      END
C*********************************************************************
      SUBROUTINE PTCOR(MODE,PTXLAT,PTYLON,PTZLEN,XPLOT,YPLOT,ZPLOT)
C*********************************************************************
C AFTER SUBROUTINE MAP HAS BEEN CALLED THIS SUBROUTINE CALLS EITHER
C                 1. ORTPLOT
C                 2. MCRPLOT
C                 3. SPRPLOT
C WHICHEVER WAS CALLED FORE MAP .THE ARGUMENTS ARE THE SAME
C AS ALL 3 OF THE ABOVE ROUTINES .
C
      REAL MODE,LATMAX,LATMIN,LONMAX,LONMIN,LATSTP,LONSTP
      INTEGER OP,DTASTP,GRID
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
C  THIS FINDS THE PROJECTED POINT .
C
      XLAT = PTXLAT
      YLON = PTYLON
      ZLEN = PTZLEN
      IF(TYP.EQ.4HORTT)
     1 CALL ORTPLOT(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C
      IF(TYP.EQ.4HMCR .OR.TYP.EQ.4HCEQA)
     1 CALL MCRPLOT(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C
      IF(TYP.EQ.4HGNM .OR.TYP.EQ.4HSTR .OR.TYP.EQ.4HAZEQ
```

```
     1 .OR.TYP.EQ.4HORTP)
     1 CALL SPRPLOT(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE PTCHECK(XPLOT,YPLOT,ZPLOT)
C*************************************************************************
C FOR EACH MAP TYPE THIS SUBROUTINE CHECKS TO SEE IF A POINT PASSES THE
C '___CHECKANDSET' TESTS FOR A POINT TO BE PLOTTED.
C IF IT DOES THEN THE XPLOT,YPLOT,ZPLOT
C VALUES ARE RETURNED AS ARGUMENTS; ALSO THE X( ),Y( ) ARRAYS IN
C COMMON/C/ ARE LOADED WITH XPLOT,YPLOT AND THE COUNTER OF POINTS TO BE
C PLOTTED 'N1'(ALSO IN COMMON/C/) WILL BE INCREMENTED.
C IF IT FAILS THE TEST THEN XPLOT,YPLOT ARE RETURNED AS 9999.
C ALSO IF IT FAILS AND :
C                    1. A PREVIOUS POINT HAS NOT FAILED
C                    2. IT WOULD NOT BE X(1),Y(1) IF LOADED
C THEN IT IS LOADED INTO X( ),Y( ) AS 9999. AND THE COUNTER 'N1' IS
C INCREMENTED .
C
      REAL MODE,LATMAX,LATMIN,LONMAX,LONMIN,LATSTP,LONSTP
      INTEGER OP,DTASTP,GRID
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
C  THIS CHECKS THE PROJECTED POINT AND DECIDES IF IT IS TO BE PLOTTED .
C
      IF(TYP.EQ.4HORTT)
     1 CALL
     1 ORTCHECKANDSET(XPLOT,YPLOT,ZPLOT)
C
      IF(TYP.EQ.4HMCR .OR.TYP.EQ.4HCEQA)
     1 CALL
     1 MCRCHECKANDSET(XPLOT,YPLOT,ZPLOT)
C
      IF(TYP.EQ.4HGNM .OR.TYP.EQ.4HSTR .OR.TYP.EQ.4HAZEQ
     1 .OR.TYP.EQ.4HORTP)
     1 CALL
     1 SPRCHECKANDSET(XPLOT,YPLOT,ZPLOT)
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE MATRIXSET
C*************************************************************************
C THIS SUBROUTINE SETS UP A MATRIX THAT CHANGES X,Y,Z COORDINATES IN AN
C EARTH COORDINATE SYSTEM(DEFINED BELOW) TO COORDINATES XX,YY,ZZ IN A
C NEW COORDINATE SYSTEM THAT HAS ITS X-AXIS POINTING TOWARDS THE
C VIEWPOINT DEFINED  BY LATITUDE 'VLAT' & LONGITUDE 'VLON' .
C
C   AN 'EARTH COORDINATE SYSTEM' IS ONE WITH THE ORIGIN AT THE
C CENTER OF THE EARTH, THE UNIT Z-VECTOR GOES TO THE NORTH POLE,
C THE UNIT X-VECTOR GOES TO THE POINT WERE THE LAT. & LON. ARE
C BOTH ZERO,AND THE UNIT Y-VECTOR IS FOUND BY FORMING THE CROSS
C PRODUCT OF UNIT Z-VECTOR INTO UNIT X-VECTOR .
C IF A POLE VIEW IS SELECTED THEN THE UNIT ECTORS CANNOT BE SET THIS
C WAY .THEN THEY ARE SET BY THE 12 LINES AFTER SUBROUTINE LINE 200 .
      DIMENSION POLE(3)
      INTEGER OP,DTASTP,GRID
      REAL LATMAX,LATMIN,LONMAX,LONMIN,LATSTP,LONSTP
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
      DATA POLE(1),POLE(2),POLE(3)/0.,0.,1./
C
C THIS SETS UP VECTORS V , RT , UP IF OP = 0   .
C
      IF( OP .NE. 0 ) GO TO 200
C
      VLEN = 1.
      CALL XYZPOLAR(4HANG ,V(1),V(2),V(3),VLAT,VLON,VLEN )
      CALL CROS1(POLE,V,RT)
```

```
      CALL CROS1(V,RT,UP)
C
      DO 100 J = 1 , 3
      AM( 1 , J ) = V( J )
      AM( 2 , J ) = RT( J )
      AM( 3 , J ) = UP( J )
  100 CONTINUE
      RETURN
  200 S = SIGN( 1. , FLOAT( OP ) )
      SP = ( 1. + S ) / 2.
      SM = ( 1. - S ) / 2.
      AM( 1 , 1 ) = 0.
      AM( 1 , 2 ) = 0.
      AM( 1 , 3 ) = S
      AM( 2 , 1 ) = SP
      AM( 2 , 2 ) = SM
      AM( 2 , 3 ) = 0.
      AM( 3 , 1 ) = SM
      AM( 3 , 2 ) = SP
      AM( 3 , 3 ) = 0.
C
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE PROJMATRIX(XIN,YIN,ZIN,XPLOT,YPLOT,ZPLOT)
C*************************************************************************
C THIS SUBROUTINE CONVERTS XYZ EARTH COORDINATES (XIN,YIN,ZIN) TO
C VIEWPOINT COORDINATES AS DESCRIBED IN SUBROUTINE MATRIXSET .
C THIS IS CALLED BY SUBROUTINES 'SPRCOR' & 'ORTCOR' .
C
C MATRIXSET MUST BE HAVE BEEN CALLED .
C
      DIMENSION PROJIN(3),PROJOUT(3)
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
C
      PROJIN(1)=XIN
      PROJIN(2)=YIN
      PROJIN(3)=ZIN
C
      CALL MLT331(PROJIN,PROJOUT,AM)
C
      XPLOT=PROJOUT(1)
      YPLOT=PROJOUT(2)
      ZPLOT=PROJOUT(3)
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE PRINTVALUES( ISLLIM , BNDOP , SHORE )
C*************************************************************************
C THIS SUBROUTINE PRINTS OUT ALL OF THE VARIABLES THAT WERE AVAILABLE
C FOR CHANGING WITH 'MORDAT' DURING EXECUTION ,IT PRINTS BOTH THOSE THAT
C HAD SOME MEANING FOR A PARTICULAR CASE AND THOSE THAT DIDN'T .
      REAL LATMIN,LATMAX,LONMIN,LONMAX,LATSTP,LONSTP,FILE700(5)
      INTEGER DTASTP,BNDOP,OP,GRID,SHORE
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
C THIS WRITES THE INPUT VARIABLES ON UNIT 700. BOTH USED AND UNUSED.
C VARIABLE VALUES ARE WRITTEN.
C
      OPEN(700,NAME='OUTDATA',STATUS='UNKNOWN')
      CLOSE(700)
      OPEN(700,NAME='OUTDATA',STATUS='OLD')
      WRITE(700,9000)
 9000 FORMAT(1H1,'  ')
C
      WRITE(700,80)
      WRITE(700,90)
```

```
   80 FORMAT(2X,'MAP DATA INPUT')
   90 FORMAT(2X,'--------------')
C
      WRITE(700,100)VLAT
      WRITE(700,200)VLON
      WRITE(700,300)LATMIN
      WRITE(700,400)LATMAX
      WRITE(700,500)LONMIN
      WRITE(700,600)LONMAX
      WRITE(700,700)TYP
      WRITE(700,800)DTASTP
      WRITE(700,1000)ISLLIM
      WRITE(700,1100)BNDOP
      WRITE(700,1200)SCALE
      WRITE(700,1400)GRID
      WRITE(700,1500)LATSTP
      WRITE(700,1600)LONSTP
      WRITE(700,1800)SHORE
C
  100 FORMAT(2X,'VLAT= ',F10.4)
  200 FORMAT(2X,'VLON= ',F10.4)
  300 FORMAT(2X,'LATMIN= ',F10.4)
  400 FORMAT(2X,'LATMAX= ',F10.4)
  500 FORMAT(2X,'LONMIN= ',F10.4)
  600 FORMAT(2X,'LONMAX= ',F10.4)
  700 FORMAT(2X,'TYP=4H',A4)
  800 FORMAT(2X,'DTASTP= ',I6)
 1000 FORMAT(2X,'ISLLIM= ',I6)
 1100 FORMAT(2X,'BNDOP= ',I6)
 1200 FORMAT(2X,'SCALE= ',F10.4)
 1400 FORMAT(2X,'GRID= ',I6)
 1500 FORMAT(2X,'LATSTP= ',F10.4)
 1600 FORMAT(2X,'LONSTP= ',F10.4)
 1800 FORMAT(2X,'SHORE= ',I6)
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE PLOTSETUP
C*************************************************************************
C THIS SUBROUTINE SETS UP THE TEKTRONICS GRAPHIC SCOPE .THE SYSTEM
C INFINITY IS SET AT 9998. SO POINTS HAVING AN X OR Y COORDINATE WITH
C MAGNITUDE GREATER THAN THIS WILL CAUSE THE GRAPHICS SCOPE TO
C 'LIFT ITS PEN' AT THE LAST GOOD POINT AND BEGIN AGAIN AT THE NEXT
C GOOD POINT.
C THE WINDOW IS ADJUSTED SO THAT EVEN IF YOU ARE WORKING WITH
C DIFFERENT 'SCALE' VALUES THE PLOT NUMBERS ALWAYS CORRESPOND TO A
C UNIT SPHERE,I.E. SCALING IS DONE NOT BY CHANGING THE NUMBERS TO BE
C PLOTTED BUT BY CHANGING THE RANGE OF NUMBERS THAT MAY BE PLOTTED.
C INCREASING THE 'SCALE' VALUE INCREASES THE RANGES PLOTTED AND
C REDUCES THE SIZE OF THE PLOT .
C
      DIMENSION XXXX(5),YYYY(5)
      INTEGER ERROR
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/SCAR/FSCALE
C
C
C THIS NEXT SECTION SETS UP THE SYSTEM SO THAT CPLOT MAY BE
C USED TO PLOT DATA POINT (PAIR) STRINGS. NUMBERS TO BE PLOTTED
C .GT. 9998. ARE AT SYSTEM INFINITY AND WILL NOT BE PLOTTED.
C
      CALL INITT(960)
      CALL BINITT
      CALL INFIN(9998.)
      CALL SLIMX(0,1023)
      CALL SLIMY(0,780)
C
C 'FSCALE' IS USED TO OVERCOME ROUNDING ERRORS IN THE GRAPHICS
C PACKAGE SO THAT A FULLSCALE ORTHOGRAPHIC PROJECTION IS A CIRCLE.
C
      FSCALE = .973
```

```
C
      DIMX = 4./3. * SCALE * FSCALE
      DIMXN = -DIMX
      CALL DLIMX(DIMXN,DIMX)
      DIMY = SCALE
      DIMYN = -DIMY
      CALL DLIMY(DIMYN,DIMY)
      CALL XLAB(0)
      CALL YLAB(0)
      CALL XFRM(0)
      CALL YFRM(0)
      CALL XNEAT(0)
      CALL YNEAT(0)
      XXXX(1)=DIMX + 1.
      YYYY(1)=DIMY + 1.
      XXXX(2)=DIMX + 1.
      YYYY(2)=DIMY + 1.
      CALL NPTS(2)
      CALL LINE(0)
      CALL CHECK(XXXX,YYYY)
      CALL DSPLAY(XXXX,YYYY)
C
C     Turn off the dialog area
      CALL DAVIS(0,ERROR)
      IF (ERROR) STOP
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE RADIANCONVERSION
C*************************************************************************
C THIS SUBROUTINE CONVERTS ALL OF THE  LATITUDE & LONGITUDE DESIGNATIONS
C FROM THE 'GLOBE' SUBROUTINES TO RADIANS FROM DEGREES .
      REAL LATMIN,LATMAX,LONMIN,LONMAX,LATSTP,LONSTP
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      DATA PI/ 3.141592654 /
C
      PIC=PI/180.
C
C CONVERSION TO RADIANS
C
      VLAT=VLAT*PIC
      VLON=VLON*PIC
      LATMIN=LATMIN*PIC
      LATMAX=LATMAX*PIC
      LONMIN=LONMIN*PIC
      LONMAX=LONMAX*PIC
      LATSTP=LATSTP*PIC
      LONSTP=LONSTP*PIC
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE VIEWPOINT
C*************************************************************************
C THIS SUBROUTINE SETS THE VIEWING LONGITUDE TO ONE HALF THE SUM OF THE
C MAXIMUM AND MINIMUM LONGITUDES .THIS WAS TO BE USED FOR ALL MAP
C PROJECTIONS CALLING 'SPRPLOT',BUT IT HAS BEEN COMMENTED OUT AND
C THE ENTIRE SUBROUTINE IS NOW SIMPLY A 'RETURN' STATEMENT . THE
C 'GLOBE' WRITE-UP HAS BEEN WRITTEN AS IF THIS IS NOT USED,ALTHOUGH THE
C CALLS TO THIS REMAIN IN THE SUBROUTINES SO IT MAY BE REINSTATED IF
C DESIRED.
C
      REAL LONMIN,LONMAX
C
C TEMPORARILY THIS HAS NO EFFECT
      RETURN
C***      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
C***     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
C***     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
C THIS ASSURES THAT  VLAT , VLON  ARE SET PROPERLY .
C
```

```
C***      IF( TYP .EQ. 4HORTT .OR. TYP .EQ. 4HMCR  .OR. TYP .EQ. 4HCEQA
C***     1 ) RETURN
C***      VLON = ( LONMAX + LONMIN ) / 2.
C
C***      RETURN
      END
C*************************************************************************
      SUBROUTINE PLOTEND
C*************************************************************************
C THIS SUBROUTINE DUMPS ANY POINTS IN THE BUFFER TO THE TEKTRONIX
C SCREEN SO THAT ALL POINTS THAT HAVE BEEN SENT TO THE SCREEN SHOW UP.
C
      CALL FINITX
      CALL TSEND
      CALL BELL
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE ORTPLOT( MODE ,XLAT ,YLON ,ZLEN ,XPLOT ,YPLOT ,ZPLOT )
C*************************************************************************
C      FOR  MODE = 4HCOR
C          XLAT,YLON,ZLEN ARE INPUT AS THE  X,Y,Z  COORDINATES.
C      FOR  MODE = 4HANG
C          XLAT,YLON,ZLEN ARE INPUT AS THE LATITUDE,LONGITUDE,ALTITUDE
C          COORDINATES .
C      XPLOT,YPLOT ARE RETURNED AS THE X,Y COORDINATES TO PLOTTED ON
C      THE GRAPHICS SCOPE. ZPLOT IS RETURNED NEGATIVE IF A POINT
C      IS ON THE SIDE OF THE GLOBE AWAY FROM THE VIEWPOINT ;THE
C      MAGNITUDE OF ZPLOT IS THE ALTITUDE .
C
      REAL MODE,LATMAX,LATMIN,LONMAX,LONMIN,LATSTP,LONSTP
      INTEGER OP,DTASTP,GRID
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/ORT/OLAT,OLON,OLEN
C
      IF( MODE .EQ. 4HCOR  )GO TO 100
      CALL XYZPOLAR( 4HANG , XX , YY , ZZ , XLAT , YLON , ZLEN )
      GO TO 200
  100 XX = XLAT
      YY = YLON
      ZZ = ZLEN
      CALL XYZPOLAR( 4HCOR , XX , YY , ZZ , XLAT , YLON , ZLEN )
  200 CONTINUE
      OLAT = XLAT
      OLON = YLON
      OLEN = ZLEN
C
      CALL PROJMATRIX( XX , YY , ZZ , XM , YM , ZM  )
C
      XPLOT = YM
      YPLOT = ZM
      ZPLOT = XM
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE ORTCHECKANDSET(XPLOT,YPLOT,ZPLOT)
C*************************************************************************
C THIS SUBROUTINE CHECKS IF 'ZPLOT' IS NEGATIVE(SIGNIFYING ERROR).
C IF IT IS THEN IT ASSIGNS  X,Y  COORDINATES  (9999.,9999.).
C SINCE THIS IS LARGER THAN THE SYSTEM INFINITY SET WITHIN
C IF SUCH A DATA PAIR APPEARS IN THE ARRAYS PLOTTED WITH 'CPLOT'
C THE EFFECT WILL BE FOR THE GRAPHICS SCOPE TO 'LIFT ITS PEN' AND
C START WITH THE NEXT PLOTTABLE POINT .FROM HERE ON CALL SUCH A
C POINT A 'BAD' POINT,AND A PLOTTABLE POINT A 'GOOD' POINT .
C IF  XPLOT,YPLOT  PASS THIS TEST THEN THEY ARE LOADED INTO THE
C ARRAYS X( ),Y( ) TO BE PLOTTED AS THEY WERE COMPUTED.
C IF  XPLOT,YPLOT  HAVE BEEN SET TO (9999.,9999.)THEY ARE LOADED INTO
C ARRAYS X( ),Y( )  UNLESS :
C    1.THEY ARE THE FIRST POINTS LOADED INTO UNPLOTTED ARRAYS X( ),Y( )
C      I.E. WOULD BE  X(1),Y(1)   .
C         OR
```

```
C     2.A BAD POINT HAS ALREADY BEEN LOADED IN THE ARRAYS.
C THIS ASSURES THAT A STRING OF POINTS TO BE PLOTTED
C DOES NOT START WITH A UNPLOTTABLE DATA POINT(CAUSES PROBLEMS),AND THAT
C LONG STRINGS OF UNPLOTTABLE POINTS DO NOT FILL THE ARRAYS X( ),Y( ).
C I.E. ONE POINT IS KEPT AS A PLACE HOLDER TO 'LIFT THE PEN' IF IT WOULD
C NOT BE THE FIRST POINT TO BE PLOTTED .
C
      REAL LATMIN,LATMAX,LONMIN,LONMAX,LATSTP,LONSTP,MODE
      INTEGER DTASTP,BNDOP,OP,Q,GRID,SHORE,U
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
      COMMON/ORT/OLAT,OLON,OLEN
      COMMON/FLAGS/IFLAG1,IFLAG2,IFLAG3,IFLAG4
C
      IF( ZPLOT .LT. 0. ) GO TO 1
C
    2 CONTINUE
      IFLAG1 = 0
      N1 = N1 + 1
      X( N1 ) = XPLOT
      Y( N1 ) = YPLOT
      RETURN
C
    1 CONTINUE
      IF( N1 .EQ. 0 )IFLAG1 = 0
      XPLOT = 9999.
      YPLOT = 9999.
      IF( IFLAG1  .EQ. 1 ) RETURN
      IF( N1 .EQ. 0 ) RETURN
      IFLAG1 = 1
      N1 = N1 + 1
      X( N1 ) = 9999.
      Y( N1 ) = 9999.
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE MCRPLOT(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C*************************************************************************
C      FOR  MODE = 4HCOR
C          XLAT,YLON,ZLEN ARE INPUT AS THE  X,Y,Z  COORDINATES.
C      FOR  MODE = 4HANG
C          XLAT,YLON,ZLEN ARE INPUT AS THE LATITUDE,LONGITUDE,ALTITUDE
C          COORDINATES .
C     XPLOT,YPLOT ARE RETURNED AS THE X,Y COORDINATES TO BE PLOTTED
C     ON THE GRAPHICS SCOPE .ZPLOT IS RETURNED AS THE NEGATIVE
C     OF THE ALTITUDE IF THE POINT IS SO CLOSE TO A POLE THAT IT
C     ESSENTIALLY MAPS TO INFINITY;OTHERWISE IT IS THE ALLTITUDE.
C     OTHERWISE IT IS RETURNED AS THE ALTITUDE.
C
      REAL LATMIN,LATMAX,LONMIN,LONMAX,MLAT,MLON,MLEN,MODE
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/MCR/MLAT,MLON,MLEN
C
      DATA A,PI/.7853981634,3.141592654/
C
C SET CONSTANTS .
C
      PIN = -PI
      CON=2./ALOG10(TAN(A+LATMAX/2.)/TAN(A+LATMIN/2.))
      CON1=2./(SIN(LATMAX)-SIN(LATMIN))
C
      XT = XLAT
      YT = YLON
      ZT = ZLEN
C
      IF( MODE .EQ. 4HCOR )GO TO 100
C
      XTLAT = XT
```

```
      YTLON = YT
      ZTLEN = ZT
C
      GO TO 200
  100 CALL XYZPOLAR( 4HCOR ,XT,YT,ZT,XTLAT,YTLON,ZTLEN)
  200 CONTINUE
      MLAT = XTLAT
      MLON = YTLON
      MLEN = ZTLEN
C
      IF(TYP .EQ. 4HMCR  .AND. XTLAT .GT. 1.56)GO TO 300
      IF(TYP .EQ. 4HMCR  .AND. XTLAT .LT. -1.56)GO TO 300
      IF(LONMIN .GE. PIN )GO TO 3
      IF(YTLON .GT. LONMAX)YTLON = YTLON - 2. * PI
    3 XPLOT=(-4./3.+8./3.*(YTLON-LONMIN)/(LONMAX-LONMIN))
      IF(TYP.EQ.4HMCR )
     1 YPLOT=(CON*ALOG10(TAN(A+XTLAT/2.)/TAN(A+LATMIN/2.))-1.)
      IF(TYP.EQ.4HCEQA)
     1 YPLOT=(CON1*(SIN(XTLAT)-SIN(LATMIN))-1.)
      ZPLOT = ZTLEN
C
      RETURN
C
  300 XPLOT = 1.
      YPLOT = 1.
      ZPLOT = -1.* ZTLEN
C
      RETURN
C
      END
C*************************************************************************
      SUBROUTINE MCRCHECKANDSET(XPLOT,YPLOT,ZPLOT)
C*************************************************************************
C THIS SUBROUTINE CHECKS IF 'ZPLOT' IS NEGATIVE(SIGNIFYING ERROR).
C IF IT IS THEN IT ASSIGNS  X,Y  COORDINATES  (9999.,9999.).
C SINCE THIS IS LARGER THAN THE SYSTEM INFINITY SET WITHIN
C IF SUCH A DATA PAIR APPEARS IN THE ARRAYS PLOTTED WITH 'CPLOT'
C THE EFFECT WILL BE FOR THE GRAPHICS SCOPE TO 'LIFT ITS PEN' AND
C START WITH THE NEXT PLOTTABLE POINT .FROM HERE ON CALL SUCH A
C POINT A 'BAD' POINT,AND A PLOTTABLE POINT A 'GOOD' POINT .
C IF 'ZPLOT' IS NOT NEGATIVE IT CHECKS TO SEE IF THE LATITUDE AND
C LONGITUDE ARE WITHIN THE LIMITS SPECIFIED BY 'LATMIN,LATMAX,LONMIN,&
C LONMAX'.IF THEY ARE NOT THEN  XPLOT,YPLOT  ARE SET TO
C (9999.,9999.) .
C IF  XPLOT,YPLOT  PASS THESE TESTS THEN THEY ARE LOADED INTO THE
C ARRAYS X( ),Y( ) TO BE PLOTTED AS THEY WERE COMPUTED.
C IF  XPLOT,YPLOT  HAVE BEEN SET TO (9999.,9999.)THEY ARE LOADED INTO
C ARRAYS X( ),Y( )  UNLESS :
C    1.THEY ARE THE FIRST POINTS LOADED INTO UNPLOTTED ARRAYS X( ),Y( )
C      I.E. WOULD BE  X(1),Y(1)   .
C         OR
C    2.A BAD POINT HAS ALREADY BEEN LOADED IN THE ARRAYS.
C THIS ASSURES THAT A STRING OF POINTS TO BE PLOTTED
C DOES NOT START WITH A UNPLOTTABLE DATA POINT(CAUSES PROBLEMS),AND THAT
C LONG STRINGS OF UNPLOTTABLE POINTS DO NOT FILL THE ARRAYS X( ),Y( ).
C I.E. ONE POINT IS KEPT AS A PLACE HOLDER TO 'LIFT THE PEN' IF IT WOULD
C NOT BE THE FIRST POINT TO BE PLOTTED .
      REAL MLAT,MLON,MLEN,LONMIN,LONMAX,LATMIN,LATMAX
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/MCR/MLAT,MLON,MLEN
      COMMON/FLAGS/IFLAG1,IFLAG2,IFLAG3,IFLAG4
C
      DATA A,PI/.7853981634,3.141592654/
      PIN = -PI
C
      IF(ZPLOT.LT.0.0)GO TO 1
C
      IF(LONMIN.GE.PIN)GO TO 3
      IF(MLON.GT.LONMAX)MLON=MLON-2.*PI
    3 IF(MLAT.LT.LATMIN.OR.MLAT.GT.LATMAX)GO TO 1
      IF(MLON.LT.LONMIN.OR.MLON.GT.LONMAX)GO TO 1
C
      IF(IFLAG2.EQ.1)GO TO 2
```

```
      IF(N1.EQ.0)GO TO 2
C CHECK IF TWO SEQUENTIAL VALID POINTS ARE ON DIFFERENT SIDES OF
C THE MAP .
      IF((XPLOT*X(N1)).LT.(-1.))GO TO 1
    2 N1 = N1 + 1
      X( N1 ) = XPLOT
      Y( N1 ) = YPLOT
      IFLAG2 = 0
      RETURN
C
    1 IF( N1 .EQ. 0 )IFLAG2 = 0
      XPLOT = 9999.
      YPLOT = 9999.
      IF( N1 .EQ. 0 )RETURN
      IF( IFLAG2 .EQ. 1 )RETURN
      IFLAG2 = 1
      N1 = N1 + 1
      X( N1 ) = 9999.
      Y( N1 ) = 9999.
C
      RETURN
C
      ENTRY N1SET(XPLOT,YPLOT,ZPLOT)
      N1 = 0
      RETURN
C
      END
C*********************************************************************
      SUBROUTINE SPRPLOT(MODE,XLAT,YLON,ZLEN,XPLOT,YPLOT,ZPLOT)
C*********************************************************************
C      FOR  MODE = 4HCOR
C          XLAT,YLON,ZLEN ARE INPUT AS THE  X,Y,Z  COORDINATES.
C      FOR  MODE = 4HANG
C          XLAT,YLON,ZLEN ARE INPUT AS THE LATITUDE,LONGITUDE,ALTITUDE
C          COORDINATES .
C      XPLOT,YPLOT ARE RETURNED AS THE X,Y COORDINATES TO PLOTTED ON
C      THE GRAPHICS SCOPE. ZPLOT IS RETURNED NEGATIVE IF A POINT
C      CANNOT BE PLOTTED USING THE PARTICULAR PROJECTION SPECIFIED
C      IN THE COMMON BLOCK(OR IF IT IS ON THE FAR SIDE OF THE GLOBE
C      FOR A PARTIAL ORTHOGRAPHIC PROJECTION)
C      ;THE MAGNITUDE OF ZPLOT IS THE ALTITUDE OF THE POINT BEING
C      TESTED .
      INTEGER OP
      REAL MULT,MODE
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/SPR/SLAT,SLON,SLEN
      COMMON/DIM/ISET
C
      DATA PI/3.141592654/
C     PIN=-PI
C
      IF( MODE .EQ. 4HCOR )GO TO 100
      CALL XYZPOLAR( 4HANG , XX , YY , ZZ , XLAT , YLON , ZLEN )
      GO TO 200
  100 XX = XLAT
      YY = YLON
      ZZ = ZLEN
      CALL XYZPOLAR( 4HCOR , XX , YY , ZZ , XLAT , YLON , ZLEN )
  200 CONTINUE
      SLAT = XLAT
      SLON = YLON
      SLEN = ZLEN
C
    5 CALL PROJMATRIX( XX , YY , ZZ ,XM,YM,ZM)
C
      ALTITUDE = ZLEN
      XM = XM / ALTITUDE
      YM = YM / ALTITUDE
      ZM = ZM / ALTITUDE
      XM = SIGN( AMIN1(1.,ABS(XM))  ,  XM )
C
C     WRITE( 2 , 2000 )XM,YM,ZM,SLAT,SLON,SLEN
```

```
C2000 FORMAT(6(2X,E9.3))
      IF(TYP.EQ.4HSTR .AND. XM.NE.-1. )MULT = 2./(1.+XM)
      IF(TYP.EQ.4HSTR .AND. XM.EQ.-1. )MULT = 1.
      IF(TYP.EQ.4HGNM .AND.XM.NE.0.0)MULT=1./XM
      IF(TYP.EQ.4HGNM .AND.XM.EQ.0.0)MULT=1.
      IF(TYP.EQ.4HORTP)MULT=1.0
      A=SQRT(YM**2+ZM**2)
      IF(TYP.EQ.4HAZEQ.AND.A.EQ.0.0)MULT=0.0
      IF(TYP.EQ.4HAZEQ.AND.A.NE.0.0)MULT=(ABS(ACOS(XM)))/A
      XPLOT=MULT*YM
      YPLOT=MULT*ZM
      IF(TYP.EQ.4HSTR .AND. XM.EQ.-1. )XPLOT = 9999.
      IF(TYP.EQ.4HSTR .AND. XM.EQ.-1. )YPLOT = 9999.
      IF(TYP.EQ.4HGNM .AND.XM.EQ.0.0)XPLOT = 9999.
      IF(TYP.EQ.4HGNM .AND.XM.EQ.0.0)YPLOT = 9999.
C
C     WRITE( 3, 1000 )XM,MULT
C1000 FORMAT(2(2X,E12.5))
      ISIGN = 1
      IF( TYP .EQ. 4HGNM .AND. XM .LE. 0 )ISIGN = -1
      IF( TYP .EQ. 4HORTP.AND. XM .LT. 0 )ISIGN = -1
      IF( TYP .EQ. 4HSTR .AND. XM .LE. -1. )ISIGN = -1
      ZPLOT = SLEN * FLOAT(ISIGN)
      IF( ISET .EQ. 0 ) RETURN
C
C
      XPLOT=(-1.+(XPLOT-HORMIN)/(HORMAX-HORMIN)*2.)
      YPLOT=(-1.+(YPLOT-VRTMIN)/(VRTMAX-VRTMIN)*2.)
      ZPLOT = ZLEN * ISIGN
C
      H=HORMAX-HORMIN
      V1=VRTMAX-VRTMIN
      SCA = H/V1
      IF(H.GE.V1)YPLOT=YPLOT/SCA
      IF(H.LT.V1)XPLOT=XPLOT*SCA
C
      RETURN
      END
C
C*****************************************************************************
      SUBROUTINE SPRCHECKANDSET(XPLOT,YPLOT,ZPLOT)
C*****************************************************************************
C THIS SUBROUTINE CHECKS IF 'ZPLOT' IS NEGATIVE(SIGNIFYING ERROR OR
C FOR ORTHOGRAPHIC ,A POINT ON THE SIDE OF THE GLOBE AWAY FROM THE
C VIEWPOINT)
C IF IT IS THEN IT ASSIGNS  X,Y  COORDINATES  (9999.,9999.).
C SINCE THIS IS LARGER THAN THE SYSTEM INFINITY SET WITHIN .
C IF SUCH A DATA PAIR APPEARS IN THE ARRAYS PLOTTED WITH 'CPLOT'
C THE EFFECT WILL BE FOR THE GRAPHICS SCOPE TO 'LIFT ITS PEN' AND
C START WITH THE NEXT PLOTTABLE POINT .FROM HERE ON CALL SUCH A
C POINT A 'BAD' POINT,AND A PLOTTABLE POINT A 'GOOD' POINT .
C IF 'ZPLOT' IS NOT NEGATIVE IT CHECKS TO SEE IF THE LATITUDE AND
C LONGITUDE ARE WITHIN THE LIMITS SPECIFIED BY 'LATMIN,LATMAX,LONMIN,&
C LONMAX'.IF THEY ARE NOT THEN  XPLOT,YPLOT  ARE SET TO
C (9999.,9999.) .
C IF  XPLOT,YPLOT  PASS THESE TESTS THEN THEY ARE LOADED INTO THE
C ARRAYS X( ),Y( ) TO BE PLOTTED AS THEY WERE COMPUTED.
C IF  XPLOT,YPLOT  HAVE BEEN SET TO (9999.,9999.)THEY ARE LOADED INTO
C ARRAYS X( ),Y( )  UNLESS :
C    1.THEY ARE THE FIRST POINTS LOADED INTO UNPLOTTED ARRAYS X( ),Y( )
C      I.E. WOULD BE  X(1),Y(1)   .
C           OR
C    2.A BAD POINT HAS ALREADY BEEN LOADED IN THE ARRAYS.
C THIS ASSURES THAT A STRING OF POINTS TO BE PLOTTED
C DOES NOT START WITH A UNPLOTTABLE DATA POINT(CAUSES PROBLEMS),AND THAT
C LONG STRINGS OF UNPLOTTABLE PLINTS DO NOT FILL THE ARRAYS X( ),Y( ).
C I.E. ONE POINT IS KEPT AS A PLACE HOLDER TO 'LIFT THE PEN' IF IT WOULD
C NOT BE THE FIRST POINT TO BE PLOTTED .
      REAL LATMIN,LATMAX,LONMIN,LONMAX
      INTEGER OP
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
      COMMON/SPR/SLAT,SLON,SLEN
      COMMON/FLAGS/IFLAG1,IFLAG2,IFLAG3,IFLAG4
```

```
      DATA A,PI/.7853981634,3.141592654/
      PIN = - PI
C
      IF( ZPLOT .LT. 0 )GO TO 1
      IF(OP.EQ.0)GO TO 6
      IF(OP.EQ.1 .AND.SLAT.LT.LATMIN)GO TO 1
      IF(OP.EQ.-1 .AND. SLAT.GT.LATMAX)GO TO 1
      IF(OP.NE.0)GO TO 2
    6 Y1=SLON
      IF(LONMIN .GT. PIN)GO TO 4
      IF(SLON .LE. LONMAX)Y1=SLON
      IF(SLON .GT. LONMAX)Y1=SLON-2.*PI
    4 IF(SLAT.LT.LATMIN.OR.SLAT.GT.LATMAX)GO TO 1
      IF(Y1.LT.LONMIN.OR.Y1.GT.LONMAX)GO TO 1
C
    2 N1 = N1 + 1
      IFLAG3 = 0
      X( N1 ) = XPLOT
      Y( N1 ) = YPLOT
      RETURN
C
    1 IF(N1.EQ.0)IFLAG3 = 0
      XPLOT = 9999.
      YPLOT = 9999.
      IF(N1.EQ.0)RETURN
      IF(IFLAG3.EQ.1)RETURN
      IFLAG3 =1
      N1=N1+1
      X(N1)=9999.
      Y(N1) = 9999.
C
      RETURN
      END
C*************************************************************************
      SUBROUTINE SETFLAGS
C*************************************************************************
C THIS ROUTINE INITIALIZES ALL FLAGS USED IN THE 'GLOBE' ROUTINES .
C THIS IS NECESSARY BEFORE ANY MAP DATA IS READ FROM THE TAPE .
      COMMON/FLAGS/IFLAG1,IFLAG2,IFLAG3,IFLAG4
      IFLAG1 = 0
      IFLAG2 = 0
      IFLAG3 = 0
      IFLAG4 = 500
      RETURN
      END
C*************************************************************************
      SUBROUTINE MATPRINT
C*************************************************************************
C THIS SUBROUTINE PRINTS OUT THE PROJECTION MATRIX USED BY MAP TYPES
C REQUIRING THE USE OF 'SPRPLOT' OR 'ORTPLOT'   .
C
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
C
      WRITE( 700 , 50 )
      WRITE( 700 , 100 )
   50 FORMAT(//2X,'PROJECTION MATRIX')
  100 FORMAT(  2X,'******************')
C
      WRITE( 700 ,200 )( (AM(I,J),J=1,3) ,I=1,3 )
  200 FORMAT(2X,3(2X,F10.6))
C
      CLOSE(700)
      RETURN
      END
C*************************************************************************
      SUBROUTINE SETN1
C*************************************************************************
C THIS SUBROUTINE RESETS 'N1' THE NUMBER OF VALID PLOT POINTS IN
C ARRAYS X( ),Y( ) TO ZERO;THIS IS DONE BEFORE ANOTHER STRING OF
C POINTS IS LOADED INTO ARRAYS X( ),Y( ) .
      COMMON/C/VLAT,VLON,LATMAX,LATMIN,LONMAX,LONMIN
     1 ,SCALE,VRTMAX,VRTMIN,HORMAX,HORMIN,TYP,OP,SCA,N1,DTASTP
     1 ,UP(3),RT(3),V(3),X(500),Y(500),LATSTP,LONSTP,GRID,AM(3,3)
```

```
      N1 = 0
      RETURN
      END
C*****************************************************
      SUBROUTINE CONNEC(IUNIT,ACAN,FILE)
      REAL*4 AUNIT
      DIMENSION FILE(5)
C     EXTERNAL ISCAN
C     AUNIT=2H
C     IF(IUNIT .LE. 0) GO TO 9000
C     IF(IUNIT .LT. 10) GO TO 10
C     IF(IUNIT .LT. 100) GO TO 20
C     IF(IUNIT .LT. 1000) GO TO 30
C     GO TO 9000
C
C10   ENCODE(4,100,AUNIT)IUNIT
C     GO TO 40
C20    ENCODE(4,101,AUNIT) IUNIT
C     GO TO 40
C30    ENCODE(4,102,AUNIT) IUNIT
C
C40    CONTINUE
C     CALL ASSGN4(FILE,ISCAN(ACAN))
C     RETURN
C
9000  CONTINUE
C     CALL BLDUFT(FILE,0,ISCAN('963'))
C     CALL ASSGN4(FILE,ISCAN('TZ '))
C     WRITE(963,9630)IUNIT
      STOP
C100   FORMAT(I1,3H   )
C101   FORMAT(I2,2H  )
C102   FORMAT(I3,1H )
C9640  FORMAT(' CONNEC...ILLEGAL IUNIT =',I10)
      END
C*****************************************************
      SUBROUTINE FINITX
      CALL BELL
      CALL TSEND
      RETURN
      END
      SUBROUTINE MLT331(VIN,VOUT,A)
C  VOUT(3) = A(3,3)*VIN(3)
      DIMENSION VIN(1),VOUT(1),A(1)
      DIMENSION TEMP(3)
      DO 10 I=1,3
10       TEMP(I) = VIN(1)*A(I)+VIN(2)*A(3+I)+VIN(3)*A(6+I)
20    VOUT(1) = TEMP(1)
      VOUT(2) = TEMP(2)
      VOUT(3) = TEMP(3)
      RETURN
C     ------------------------------------------------------------------
      ENTRY MLT133(VIN,VOUT,A)
C  VOUT(1,3) = VIN(1,3)*A(3,3)
C        OR
C  VOUT(3) = AT(3,3)*VIN(3)
      JJ = 0
      DO 30 J=1,3
         TEMP(J) = A(JJ+1)*VIN(1)+A(JJ+2)*VIN(2)+A(JJ+3)*VIN(3)
30       JJ = 3+JJ
      GO TO 20
      END
      SUBROUTINE VCROSS(A,B,C)
C  C IS CROSS-PRODUCT OF A AND B
      DIMENSION A(3),B(3),C(3),TEMP(2)
      LOGICAL KROSS
      ENTRY CROSS(A,B,C)
      KROSS = .TRUE.
      GO TO 10
      ENTRY CROS1(A,B,C)
      KROSS = .FALSE.
10    CONTINUE
      TEMP(1) = A(2)*B(3)-A(3)*B(2)
      TEMP(2) = A(3)*B(1)-A(1)*B(3)
      C(3) = A(1)*B(2)-A(2)*B(1)
```

```
C(1) = TEMP(1)
C(2) = TEMP(2)
IF(KROSS) RETURN
C1 = SQRT(C(1)**2+C(2)**2+C(3)**2)
C(1) = C(1)/C1
C(2) = C(2)/C1
C(3) = C(3)/C1
RETURN
END
```

```
$!-----------------------------------------------
$!
$! Procedure  S G P L I N K
$!
$!   Date:           Author:
$!
$! September 1985   Mark Lovell
$!
$!-----------------------------------------------
$!
$!  Description:
$!      This procedure links all of the modules
$!      needed to run the SGPSAT program
$!
$!   Linking Sequence:
$!       @sgplink
$!
$!-----------------------------------------------
$
$    !SGPLINK
$
$Link sgpsat,solarvariables,blocksgp,sgpred,modjul,sgp,sgpdsp,-
  sgptim,timdsp,rdate,sgprun,sgphdr,sgpfmt,ecoec1,sgplst,sgplbl,-
  graphics,graphics1,sgpredfile,hourangle,screen,sgpinput,tscreen
$
$    !END of SGPLINK]
```

```
C
C*****************************************************************
C
C  Main Progam    S G P S A T
C
C     Date          Author
C
C   March 1980     Roy McPherson
C
C*****************************************************************
C
C Description:
C      This program simulates earth satellites
C
C*****************************************************************
C
C...Declarations
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 SUM,FLNAME(26),IC,ICM,FLAG,ERROR
      DIMENSION IOT(6)
        INTEGER BUF(80),I,VERSION_NUM
      COMMON/SUMMRY/SUM(80,12),NL
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/SGPELM/SGPDAT(14)
      DATA IC/"40,"33,"131,"65,"40,"33,"112,"40/
C
C...Executable Code
C
C...Initialize Parameters
      IN=5
      IO=6
      PI=3.1415926536D0
      AE=6378.145D0
C
      DO 5 I=1,14
5     SGPDAT(I)=0.D0
      DO 10 I=1,3
10    TL(I)=0.D0
      ICON(1)=1
      ICON(2)=1
      ICON(3)=1
      ICON(4)=0
      DO 50 I=1,8
50    ELM(I)=0.D0
      TM(1)=0.D0
      TM(2)=0.D0
      EDT=0.D0
C
C...Change to Tek mode to set up the display
C
      CALL SELECT_CODE(0)
25      CALL PAGE
        CALL DIALOG_LINES(32)
C
C...Set terminal to ansii mode
C
        CALL SELECT_CODE(1)
C
C...Clear the dialog area
C
      CALL ED(2,ERROR)
        IF (ERROR) STOP
C
C...Cursor Position At Home
C
        CALL ABSOLUTE
        CALL CUP(1,1)
C
C...Write out the program introduction to the terminal
C
        OPEN(UNIT=10,FILE='SGPINS.TXT',STATUS='OLD',
     1   IOSTAT=IOS)
```

```
        IF (IOS .NE. 0) THEN
          WRITE(6,100)IOS
100       FORMAT(' ERROR IN OPENING SGPINS, IOS IS ',I3)
          GO TO 20
        ENDIF
6       READ(10,101,END=7)(BUF(I),I=1,80)
101     FORMAT(80A1)
        WRITE(6,102)(BUF(I),I=1,80)
102     FORMAT(' ',80A1)
        GO TO 6
7       CONTINUE
C
505     FORMAT(80A1)
C
C...Read In The Epoch Conditions
C
      WRITE(ID,506)
506   FORMAT('$','IS THE DATA ON FILE ? (Y)=YES )')
      READ(IN,505) ICM
C
C...Read Data File Or From The Screen
C....If FLAG = 1, Read From File. If FLAG = 0, Read From Screen
C
      FLAG = 0
      IF (ICM.EQ.'Y' .OR. ICM.EQ.'y') FLAG = 1
C
C...File Reading
C
      IF (FLAG.EQ.1)THEN
C
C...Read The File
C
            CALL SGPREDFILE(ERROR)
C
C....If Error In File, End Or Read From Screen
C
            IF (ERROR)THEN
              WRITE(ID,507)
507         FORMAT('$DO YOU WANT TO ENTER THE DATA NOW? (Y)=YES ')
            READ(IN,505) ICM
              IF(ICM.EQ.'Y' .OR. ICM.EQ.'y') THEN
C.....Read From Screen
                FLAG = 0
            ELSE
C.....Exit
                CALL SELECT_CODE(0)
                  GO TO 20
              ENDIF
            ENDIF
       ENDIF
C
C...Screen Entering
C
      IF (FLAG.EQ.0)THEN
C
C....Select Ansii Mode For Data Display Only
C
            CALL SELECT_CODE(1)
          CALL SGPINPUT
          CALL SGPRED
      ENDIF
C
C...Select Tek Mode And Proceed With The Simulation
C
C...Copy Simulation Info To File
C
      OPEN(UNIT = 9,FILE = 'SIMDAT.DAT',STATUS = 'NEW', IOSTAT = IOS)
      IF (ICON(1) .EQ. 1) WRITE(9,509) TL(3)
      WRITE(9,508) (SGPDAT(I),I=1,7)
508   FORMAT(1X,7F12.6)
509     FORMAT(1X,F9.3)

            CALL SELECT_CODE(0)
C
C...Restrict the dialog area to two lines
```

```
C
        CALL DIALOG_LINES(2)
C
      WRITE(IO,515)
515   FORMAT('$',' RUN THE SIMULATION? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.NE.'Y' .AND. ICM.NE.'y') GO TO 15
C
C...Clear the graphics area, then clear and enlarge the dialog area
C
        CALL PAGE
        CALL DIALOG_LINES(32)
        CALL CLEAR_DIALOG_AREA
C
C...Run The Simulation
C
      WRITE(IO,520)
520   FORMAT(' SIMULATION RUNNING')
      CALL SGPRUN
C
C...A Source Data File Called SGPSIM.DAT Has Been Produced,
C   And Is Logical UNIT 17 (Rewound And Ready For Further Access)
C
C...Calculate Apogee/Perigee & Equatorial crossing Times?
C
      WRITE(IO,575)
575   FORMAT('$',' CALCULATE APOGEE/PERIGEE & CROSSING TIMES?',
     1 ' (Y)=YES )')
      READ(IN,505) ICM
      IOT(1)=0
      IF(ICM.NE.'Y' .AND. ICM.NE.'y')GO TO 55
      IOT(1)=1
      WRITE(IO,580)
580   FORMAT(' TIMING DATA BEING CALCULATED')
      CALL SGPTIM(17)
C
55    WRITE(IO,595)
595   FORMAT(' PROGRAM IS BUSY')
      ENCODE(14,590,FLNAME) ICON(1)
590   FORMAT('SGPSIM.DMP;',03)
C     CALL ASSIGN(7,FLNAME,14)
        OPEN(UNIT=7,FILE=FLNAME,STATUS='UNKNOWN')
      CALL SGPHDR(7,IOT)
C
C...Request For Output Listings
C
      WRITE(IO,585)
585   FORMAT('$',' DO YOU WANT OUTPUT LISTINGS? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.NE.'Y' .AND. ICM.NE.'y') GO TO 15
C
      DO 30 I=2,6
30    IOT(I)=0
      WRITE(IO,525)
525   FORMAT('$',' DO YOU WANT INERTIAL CARTESIAN? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.EQ.'Y' .OR. ICM.EQ.'y') IOT(2)=1
      WRITE(IO,535)
535   FORMAT('$',' DO YOU WANT EARTH CARTESIAN? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.EQ.'Y' .OR. ICM.EQ.'y') IOT(3)=1
      WRITE(IO,540)
540   FORMAT('$',' DO YOU WANT GEOGRAPHIC FORMAT? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.EQ.'Y' .OR. ICM.EQ.'y') IOT(4)=1
      WRITE(IO,545)
545   FORMAT('$',' DO YOU WANT ORBITAL FORMAT? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.EQ.'Y' .OR. ICM.EQ.'y') IOT(5)=1
C
      TREL=0.D0
      WRITE(IO,550)
550   FORMAT('$',' DO YOU WANT A RELATIVE TIME SCALE ON LISTINGS'
     1 ,' (Y)=YES )')
      READ(IN,505) ICM
```

```
      IF(ICM.NE.'Y' .AND. ICM.NE.'y') GO TO 45
      IOT(6)=1
      WRITE(IO,560)
560   FORMAT(' OFFSET TIME(MINS), RELATIVE TO SIMULATION RUNNING ',
     1 'TIME',/,'$)')
      READ(IN,565) TREL
565   FORMAT(D25.16)
C
C...Open The Display Files
C
45    DO 35 I=1,4
      IF(IOT(I+1).EQ.0) GO TO 35
      VERSION_NUM = I*100 + ICON(1)
      ENCODE(14,570,FLNAME) VERSION_NUM
570   FORMAT('SGPSAT.DSP;',I3)
        OPEN(UNIT=I,FILE=FLNAME,STATUS='UNKNOWN')
35    CONTINUE
C
C...Listings Preformatter Routine
C
      WRITE(IO,555)
555   FORMAT(' LIST FILE BEING MADE FOR THE SPOOLER')
      CALL SGPFMT(17,IOT,TREL)
      CALL SGPLST(7,IOT)
C
C...Close The Display Files
C
      DO 40 I=1,4
      IF(IOT(I+1).EQ.0) GO TO 40
      CALL CLOSE(I)
40    CONTINUE
C
15      CONTINUE
C     CALL SPOOL(7,ISTAT)
      CALL CLOSE(7)
      CALL CLOSE(17)
      WRITE(IO,530)
530   FORMAT('$',' ANOTHER RUN? (Y)=YES )')
      READ(IN,505) ICM
      IF(ICM.EQ.'Y' .OR. ICM.EQ.'y') THEN
        ICON(1)=ICON(1)+1
        GO TO 25
      ENDIF
C
C...Clear the screen
C
20    CALL CLEAR_DIALOG_AREA
C
C...Change to Ansii mode
C
        CALL PAGE
        CALL DIALOG_LINES(32)
        CALL SELECT_CODE(1)
C
C
C...Exit
      CALL EXIT
      END
```

```
#include math

/*-------------------------------------------------------------------------
 *  Routines   S O L A R   V A R I A B L E S
 *
 *
 *
 *    Date:              Author:
 *  September 27, 1985  Mark W. Lovell
 *
 *-------------------------------------------------------------------------
 *
 *    Description:
 *      This routine calculates the sun's time constants and orbital
 *       parameters for the given date as in 'Orbital Determination and
 *       Prediction Mathematics' - Section 7 - 'EPHEMERIS'
 *       NOTE : angles in degrees
 *
 *    Calling Sequence:
 *
 *     FUNCTION_NAME( Time )
 *     Time       : Time (in Modified Julian) for which to calculate sun's
 *                  position - time variable must be a 'double' precision
 *                      variable.
 *-------------------------------------------------------------------------
 *

   ----------------JULIAN TIME-------------------------

    FUNCTION - julian_days - time in ephemeris days - page 7-5 */

double julian_days( time )
  double time;
 {
   return(-15019.5008338 + 1.00000003201*time);
  }


 /* FUNCTION - julian_centuries - time in Julian centuries - page 7-5 */

double julian_centuries( time )
  double time;
 {
   return( julian_days(time)/36525.0 );
  }


/*----------------SOLAR PARAMETERS----------------------*/

 /* FUNCTION - sun_w_deg - longitude of perigee (consider the sun as a
                           satellite) from page 7-3 */

double sun_w_deg( time )
  double time;
 {
   double d;
   double T;

   d = julian_days(time);
   T = julian_centuries( time );

   return(281.22083 + 0.0000470684*d + 0.000453*T*T + 0.000003*T*T*T );

  }


 /* FUNCTION - sun_M_deg - Mean anomoly - from page 7-3 */

double sun_M_deg( time )
  double time;
 {
   double d;
```

```
    double T;

    d = julian_days(time);
    T = julian_centuries( time );

    return( 358.47583 + 0.985600267*d - 0.00015*T*T - 0.000003*T*T*T );

   }


 /* FUNCTION - sun_epsilon - Mean Obliquity- from page 7-5 */

double sun_epsilon( time )
  double time;
 {
   double T;

   T = julian_centuries( time );

   return( 23.452294 - 0.0130125*T - 0.00000164*T*T + 0.000000503*T*T*T );

  }


 /* FUNCTION - sun_eccentric - Sun's eccentricity - from page 7-3 */

double sun_eccentric( time )
  double time;
 {
   double T;

   T = julian_centuries( time );

   return( 0.01675104 - 0.0000418*T - 0.000000126*T*T );

  }


 /* FUNCTION - delta_psi - Nutation in longitude - from page 7-9 */

double delta_psi( time )
  double time;
 {
   double d;
   double F,D,OHM;
   double sunf(),sund(),sunohm();  /*Constants - see SUN CONSTANTS */
   double dtr = 3.1415926536/180;  /* degree to radian convertion */
   double dw;

     d = julian_days(time);

     F = sunf(d)*dtr;
     D = sund(d)*dtr;
     OHM = sunohm(d)*dtr;


     dw = -17.2327*sin(OHM) - 1.2729*sin( 2*(F-D+OHM) );
     dw = dw + 0.2088*sin(2*OHM) - 0.2037* sin( 2*(F+OHM) );

     dw = dw/3600;  /* convert to degrees */

   return( dw );

  }


 /* FUNCTION - delta_epsilon - Nutation in obliquity - from page 7-9 */

double delta_epsilon( time )
  double time;
 {
   double d;
```

```
    double F,D,OHM;
    double sunf(),sund(),sunohm();  /*Constants - see SUN CONSTANTS */
    double dtr = 3.1415926536/180;  /* degree to radian convertion */
    double de;

      d = julian_days(time);

      F = sunf(d)*dtr;
      D = sund(d)*dtr;
      OHM = sunohm(d)*dtr;

      de = 9.21* cos(OHM) + 0.5522* cos( 2*(F-D+OHM) );

      de = de/3600;  /*convert from seconds to degrees */

    return( de );

  }


/*--------------------SUN CONSTANTS--------------------------*/

/* Solve for 'dw' ( nutation in longitude ) and 'de' ( nutation in obliquity )
   as in Section 7.5 by finding three constants, 'F', 'D', and 'OHM' */

 /* FUNCTION - sunf - Sun constant 1 - from page 7-9 */

double sunf( days )
  double days;
 {
  return( 11.250889 + 13.2293504490*days - 2.407e-12*days*days );
  }


 /* FUNCTION - sund - Sun constant 2 - from page 7-9 */

double sund( days )
  double days;
 {
  return( 350.737486 + 12.1907491914*days - 1.076e-12*days*days );

  }


 /* FUNCTION - sunohm - Sun constant 3 - from page 7-9 */

double sunohm( days )
  double days;
 {
  return( 259.183275 - 0.0529539222*days + 1.557e-12*days*days );

  }
```

```
C
C*****************************************************************
C
C  Common Blocks    B L O C K S G P
C
C     Date          Author
C
C   Sept. 1985     Suzanne Y. Slinn
C
C*****************************************************************
C
C Description:
C      This program defines all the common blocks
C
C*****************************************************************
C
C...Declarations
C
       Block Data X
       Implicit Real*8(A-H,O-Z)
       integer STORE/0/
       Logical*1 SUM(80,12)
C
C...Executable Code
C
       COMMON/SUMMRY/SUM,NL
       COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
       COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
       COMMON/SGPELM/SGPDAT(14)
       COMMON/TIMSGP/L,N(30),TMS(3,30)
       COMMON/CHANNEL/STORE
       END
```

```
C
C*****************************************************************
C
C  Subroutine   S G P R E D
C
C     Date          Author
C
C   Oct 15,1980     Mark Lovell
C
C*****************************************************************
C
C Description:
C      This routine is a screen editor for entering simulation
C        parameters.
C
C Calling Sequence :
C      Call Sgpred
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPRED
      IMPLICIT COMPLEX*16(A-Z)
      INTEGER IN,IO,ICON
      INTEGER IYR,MNTH,IDY,IZULU,IDIR,I,NC,IERR
      INTEGER CODE
      REAL RMON(12)
      REAL*8 PI,AE,ELM,TM,EDT,TL,SGPDAT
      REAL*8 DZULU,ZULU,TEMP
      LOGICAL*1 BDAT(10),IC,IC1
      CHARACTER*1 NULL/'0'X/
      DIMENSION DREV(3),POSN(6)
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/SGPELM/SGPDAT(14)
      DATA RMON/'JAN ','FEB ','MAR ','APR ','MAY ','JUN ',
     1 'JUL ','AUG ','SEP ','OCT ','NOV ','DEC '/
C
C...Executable Code
C
C
10    WRITE(IO,20)
20    FORMAT('$','DO YOU WANT TO CHANGE ANY PARAMETERS? (Y)=YES )')
      READ(IN,560) IC1
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C
      IF(IC1.NE.'Y' .AND. IC1.NE.'y') GO TO 9000
C
C...Screen Editing Mode
C
      CALL CUP(22,1)
      WRITE(IO,30)NULL
30    FORMAT(A1,' ENTER DATA IN REAL FORMAT')
      WRITE(IO,35)
35    FORMAT('  F1,F2,F3,F4 - SELECT DATA ITEM')
      WRITE(IO,36)
36    FORMAT('  F5 - SIGNIFIES DATA ENTRY')
      WRITE(IO,37)
37    FORMAT('  F8 - EXIT FROM EDITOR')
C
C...Entering Of Epoch Date
C....Standard Format
C
C....Enhance Title
C
40    CALL TITLE(5,13,'EPOCH',1)
      CALL TITLE(3,30,'DATE',1)
C
C...Cursor At Data Line
C
```

```
45    CALL CUP(5,28)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (46,48,46,48,50,50,9000,9000), CODE
C
C...Go To Previous Data Item
C
46    CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,30,'DATE',0)
      GO TO 1540
C
C...Go To Next Data Item
C
48    CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,30,'DATE',0)
      GO TO 140
C
C...Get This Data Item
C....Erase Old Data And Print It Below
C
50    CALL CUP(5,29)
      WRITE(IO,52)NULL
52    FORMAT(A1,'          ')
C
      IDIR = 1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
      CALL CUP(6,29)
      WRITE(IO,54) NULL,IDY,RMON(MNTH),IYR
54    FORMAT(A1,I2,1X,A4,I2)
C
C
C...Read New Item
C
      CALL CUP(5,29)
      READ(IN,56) NC,(BDAT(I),I=1,NC)
56    FORMAT(Q,9A1)
C
C....Check Input For A Null Entry, If So Go To Next Data Item
C
      IF (NC .EQ. 0) GO TO 70
C
      CALL RDATE(BDAT,NC,IDY,MNTH,IYR,IERR)
C
C...Error Message
C
      IF(IERR) THEN
        CALL BELL
        CALL TITLE(28,5,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER DY MTH YR -EG. 22 APR 65',1)
        GO TO 50
      ENDIF
C
C...Compute Julian Date And Display It
C
      IDIR=0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
C
C....Blank Out Old Julian Date And Print New
C
      CALL CUP(5,54)
      WRITE(IO,60)NULL
60    FORMAT(A1,'          ')
      CALL CUP(5,54)
      WRITE(IO,65)NULL,TM(2)
65    FORMAT(A1,F12.6)
C
C....Erase Old Data And Print A Good Version Of New Data
C
70    CALL CUP(5,29)
      WRITE(IO,72)NULL
```

```
      CALL CUP(6,29)
      WRITE(IO,72)NULL
72    FORMAT(A1,'          ')
C
      CALL CUP(5,29)
      WRITE(IO,74) NULL,IDY,RMON(MNTH),IYR
74    FORMAT(A1,I2,1X,A4,I2)

C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,30,'DATE',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C
C

C...Entering OF Epoch Zulu Time
C
C....Enhance Title
C
140   CALL TITLE(5,13,'EPOCH',1)
      CALL TITLE(3,41,'TIME',1)
C
C...Cursor At Data Line
C
145   CALL CUP(5,41)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (146,148,146,148,150,150,9000,9000), CODE
C
C...Go To Previous Data Item
C
146   CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,41,'TIME',0)
      GO TO 40

C...Go To Next Data Item
C
148   CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,41,'TIME',0)
      GO TO 240
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
150   IDIR = 1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
      CALL CUP(6,42)
      WRITE(IO,151) NULL,ZULU
151   FORMAT(A1,F9.3)
C
152   CALL CUP(5,42)
      WRITE(IO,154)NULL
154   FORMAT(A1,'          ')
C
C
C...Read New Data Item
C
      CALL CUP(5,43)
      READ(IN,155,ERR=156) ZULU
155   FORMAT(F8.3)
C
      GO TO 159
C
C...Error Message
C
```

```
156     CALL BELL
        CALL TITLE(28,13,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'HHMM.MMM -EG. 2330.500 = 11:30 PM AND 30 SECONDS'
     +          ,1)
        GO TO 152
C
C...Compute Julian Date And Display It
C
159   IDIR=0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
C
C....Blank Out Old Date And Print New
C
      CALL CUP(5,54)
      WRITE(IO,160)NULL
160   FORMAT(A1,'              ')
      CALL CUP(5,54)
      WRITE(IO,165)NULL,TM(2)
165   FORMAT(A1,F12.6)
C
C....Erase Old Data And Print A Good Version Of New Data
C
170   CALL CUP(5,42)
      WRITE(IO,172)NULL
      CALL CUP(6,42)
      WRITE(IO,172)NULL
172   FORMAT(A1,'          ')
C
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
        IZULU = ZULU/100
          DZULU = ZULU - (IZULU*100)
C
      CALL CUP(5,42)
      WRITE(IO,174) NULL,IZULU,DZULU
174   FORMAT(A1,I2,1X,F6.3)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,41,'TIME',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Epoch Date
C....Julian Format
C
C...Enhance Title
C
240   CALL TITLE(5,13,'EPOCH',1)
      CALL TITLE(3,53,'JULIAN DATE',1)
C
C...Cursor At Data Line
C
245   CALL CUP(5,53)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (246,248,246,248,250,250,9000,9000), CODE
C
C...Go To Previous Data Item
C
246   CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,53,'JULIAN DATE',0)
      GO TO 140
C
C...Go To Next Data Item
```

```
C
248   CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,53,'JULIAN DATE',0)
      GO TO 340
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
250   CALL CUP(6,54)
      WRITE(IO,251) NULL,TM(2)
251   FORMAT(A1,F12.6)
C
252   CALL CUP(5,54)
      WRITE(IO,253)NULL
253   FORMAT(A1,'            ')
C
C...Read New Data Item
C
254   CALL CUP(5,54)
      READ(IN,255,ERR=256) TEMP
255   FORMAT(F12.6)
C
C....Check Data For A Null Entry, If So Go To Next Data Item
C
      IF(TEMP .EQ. 0.D0) GO TO 270
      IF(TEMP .LT. 33282.D0) GO TO 256
C
C...If Entry Is A Valid Number, Assign It To TM(2)
C
      TM(2)=TEMP
      GO TO 259
C
C...Error Message
C
256     CALL BELL
        CALL TITLE(28,13,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'DDDDD.DDDDDD -EG. 43509.000000 = JAN 1 1978 AT 0h'
     +            ,1)
        GO TO 252
C
C...Compute Standard Date And Time And Display It
C
259   IDIR=1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
        IZULU = ZULU/100
          DZULU = ZULU - (IZULU*100)
C
C
C....Blank Out Old Date And Print New
C
      CALL CUP(5,29)
      WRITE(IO,260)NULL
260   FORMAT(A1,'                         ')
      CALL CUP(5,29)
      WRITE(IO,265) NULL,IDY,RMON(MNTH),IYR,IZULU,DZULU
265   FORMAT(A1,I2,1X,A4,I2,4X,I2,1X,F6.3)
      IDIR = 1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
C
C....Erase Old Data And Print A Good Version Of New Data
C
270   CALL CUP(5,54)
      WRITE(IO,272)NULL
      CALL CUP(6,54)
      WRITE(IO,272)NULL
272   FORMAT(A1,'              ')
C
      CALL CUP(5,54)
      WRITE(IO,274) NULL,TM(2)
274   FORMAT(A1,F12.6)

C
```

```
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(5,13,'EPOCH',0)
      CALL TITLE(3,53,'JULIAN DATE',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C
C...Entering Of Simulation Date
C....Standard Format
C
C...Enhance Title
C
340   CALL TITLE(7,13,'SIMULATION',1)
      CALL TITLE(3,30,'DATE',1)
C
C...Cursor At Data Line
C
345   CALL CUP(7,28)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (346,348,346,348,350,350,9000,9000), CODE
C
C...Go To Previous Data Item
C
346   CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,30,'DATE',0)
      GO TO 240
C
C...Go To Next Data Item
C
348   CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,30,'DATE',0)
      GO TO 440
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
350   CALL CUP(7,29)
      WRITE(IO,352)NULL
352   FORMAT(A1,'          ')
C
      IDIR = 1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
      CALL CUP(8,29)
      WRITE(IO,354) NULL,IDY,RMON(MNTH),IYR
354   FORMAT(A1,I2,1X,A4,I2)
C
C...Read New Data Item
C
      CALL CUP(7,29)
      READ(IN,356) NC,(BDAT(I),I=1,NC)
356   FORMAT(Q,9A1)
C
C....Check Input For A Null Entry. If So Go To Next Data Item
C
      IF (NC .EQ. 0) GO TO 370
C
      CALL RDATE(BDAT,NC,IDY,MNTH,IYR,IERR)
C
C...Error Message
C
      IF(IERR) THEN
        CALL BELL
        CALL TITLE(28,5,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER DY MTH YR -EG. 22 APR 65',1)
        GO TO 350
      ENDIF
```

```
C
C...Compute Julian Date And Display It
C
      IDIR=0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
C
C....Blank Out Old Date And Print New
C
      CALL CUP(7,54)
      WRITE(IO,360)NULL
360   FORMAT(A1,'              ')
      CALL CUP(7,54)
      WRITE(IO,365)NULL,TM(1)
365   FORMAT(A1,F12.6)
C
C....Erase Old Data And Print A Good Version Of New Data
C
370   CALL CUP(7,29)
      WRITE(IO,372)NULL
      CALL CUP(8,29)
      WRITE(IO,372)NULL
372   FORMAT(A1,'           ')
C
      CALL CUP(7,29)
      WRITE(IO,374) NULL,IDY,RMON(MNTH),IYR
374   FORMAT(A1,I2,1X,A4,I2)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,30,'DATE',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C
C
C...Entering Of Simulation Zulu Time
C
C....Enhance Title
C
440   CALL TITLE(7,13,'SIMULATION',1)
      CALL TITLE(3,41,'TIME',1)
C
C...Cursor At Data Line
C
445   CALL CUP(7,41)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (446,448,446,448,450,450,9000,9000), CODE
C
C...Go To Previous Data Item
C
446   CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,41,'TIME',0)
      GO TO 340
C
C...Go To Next Data Item
C
448   CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,41,'TIME',0)
      GO TO 540
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
450   IDIR = 1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
      CALL CUP(8,42)
```

```
      WRITE(ID,451) NULL,ZULU
451   FORMAT(A1,F9.3)
C
452   CALL CUP(7,42)
      WRITE(ID,453)NULL
453   FORMAT(A1,'          ')
C
C
C...Read This Data Item
C
454   CALL CUP(7,43)
      READ(IN,455,ERR=456) ZULU
455   FORMAT(F8.3)
C
      GO TO 459
C
C...Error Message
C
456     CALL BELL
        CALL TITLE(28,13,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'HHMM.MMM -EG. 2330.500 = 11:30 PM AND 30 SECONDS'
     +          ,1)
        GO TO 452
C
C...Compute Julian Date And Display It
C
459   IDIR=0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
C
C....Blank Out Old Date And Print New
C
      CALL CUP(7,54)
      WRITE(ID,460)NULL
460   FORMAT(A1,'          ')
      CALL CUP(7,54)
      WRITE(ID,465)NULL,TM(1)
465   FORMAT(A1,F12.6)
C
C....Erase Old Data And Print A Good Version Of New Data
C
470   CALL CUP(7,42)
      WRITE(ID,472)NULL
      CALL CUP(8,42)
      WRITE(ID,472)NULL
472   FORMAT(A1,'          ')
C
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
        IZULU = ZULU/100
          DZULU = ZULU - (IZULU*100)
C
      CALL CUP(7,42)
      WRITE(ID,474) NULL,IZULU,DZULU
474   FORMAT(A1,I2,1X,F6.3)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,41,'TIME',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Simulation Date
C....Julian Format
C
C...Enhance Title
C
540   CALL TITLE(7,13,'SIMULATION',1)
      CALL TITLE(3,53,'JULIAN DATE',1)
C
C...Cursor At Data Line
```

```
C
545   CALL CUP(7,53)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (546,548,546,548,550,550,9000,9000), CODE
C
C...Go To Previous Data Item
C
546   CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,53,'JULIAN DATE',0)
      GO TO 440
C
C...Go To Next Data Item
C
548   CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,53,'JULIAN DATE',0)
      GO TO 640
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
550   CALL CUP(8,54)
      WRITE(IO,551) NULL,TM(1)
551   FORMAT(A1,F12.6)
C
552   CALL CUP(7,54)
      WRITE(IO,553)NULL
553   FORMAT(A1,'             ')
C
C
C...Read This Data Item
C
554   CALL CUP(7,54)
      READ(IN,555,ERR=556) TEMP
555   FORMAT(F12.6)
C
C
C...Check Data For A Null Entry, If So Go To Next Data Item
C
      IF(TEMP .EQ. 0.D0) GO TO 570
      IF(TEMP .LT. 33282.D0) GO TO 556
C
C...If Entry Is A Valid Number, Assign It To TM(1)
C
      TM(1)=TEMP
      GO TO 559
C
C...Error Message
C
556     CALL BELL
        CALL TITLE(28,13,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'DDDDD.DDDDDD -EG. 43509.000000 = JAN 1 1978 AT 0h'
     +             ,1)
        GO TO 552
C
C...Compute Standard Date And Time And Display It
C
559   IDIR=1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
        IZULU = ZULU/100
          DZULU = ZULU - (IZULU*100)
C
C
C...Blank Out Old Date And Print New
C
      CALL CUP(7,29)
      WRITE(IO,560)NULL
```

```
560   FORMAT(A1,'                    ')
      CALL CUP(7,29)
      WRITE(IO,565) NULL,IDY,RMON(MNTH),IYR,IZULU,DZULU
565   FORMAT(A1,I2,1X,A4,I2,4X,I2,1X,F6.3)
C
C....Erase Old Data And Print A Good Version Of New Data
C
570   CALL CUP(7,54)
      WRITE(IO,572)NULL
      CALL CUP(8,54)
      WRITE(IO,572)NULL
572   FORMAT(A1,'            ')
C
      CALL CUP(7,54)
      WRITE(IO,574) NULL,TM(1)
574   FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(7,13,'SIMULATION',0)
      CALL TITLE(3,53,'JULIAN DATE',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Simulation Run Time
C
C...Enhance Title
C
640   CALL TITLE(9,6,'TOTAL RUN(MINS)',1)
C
C...Cursor At Data Line
C
645   CALL CUP(9,25)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (646,648,646,648,650,650,9000,9000), CODE
C
C...Go To Previous Data Item
C
646   CALL TITLE(9,6,'TOTAL RUN(MINS)',0)
      GO TO 540
C
C...Go To Next Data Item
C
648   CALL TITLE(9,6,'TOTAL RUN(MINS)',0)
      GO TO 740
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
650   CALL CUP(10,26)
      WRITE(IO,651) NULL,TL(2)
651   FORMAT(A1,F9.3)
C
652   CALL CUP(9,26)
      WRITE(IO,654)NULL
654   FORMAT(A1,'          ')
C
C...Read New Data Item
C
      CALL CUP(9,27)
      READ(IN,655,ERR=656) TEMP
655   FORMAT(F9.3)
C
C...Check Entry For A Valid Number, Assign It To TL(2)
C
      IF(TEMP .NE. 0.D0) TL(2) = TEMP
```

```
      GO TO 670
C
C.,.Error Message
C
656      CALL BELL
         CALL TITLE(28,13,'ERROR IN FORMAT, TRY AGAIN',1)
         CALL TITLE(29,3,'MMMMM.MMM -EG 01000.000 = 1000 MIN SIMULATION',1)
         GO TO 652
C
C...Erase Old Data And Print A Good Version Of New Data
C
670   CALL CUP(9,26)
      WRITE(IO,672)NULL
      CALL CUP(10,26)
      WRITE(IO,672)NULL
672   FORMAT(A1,'          ')
C
      CALL CUP(9,26)
      WRITE(IO,674) NULL,TL(2)
674   FORMAT(A1,F9.3)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(9,6,'TOTAL RUN(MINS)',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Simulation Delta Time
C
C...Enhance Title
C
740   CALL TITLE(9,42,'DELTA TIME(MINS)',1)
C
C...Cursor At Data Line
C
745   CALL CUP(9,63)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (746,748,746,748,750,750,9000,9000), CODE
C
C...Go To Previous Data Item
C
746   CALL TITLE(9,42,'DELTA TIME(MINS)',0)
      GO TO 640
C
C...Go To Next Data Item
C
748   CALL TITLE(9,42,'DELTA TIME(MINS)',0)
      GO TO 840
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
750   CALL CUP(10,63)
      WRITE(IO,751) NULL,TL(3)
751   FORMAT(A1,F9.3)
C
752   CALL CUP(9,63)
      WRITE(IO,754)NULL
754   FORMAT(A1,'          ')
C
C...Read New Data Item
C
      CALL CUP(9,64)
      READ(IN,755,ERR=756) TEMP
755   FORMAT(F9.3)
C
```

```
C...Check Entry For A Valid Number. Assign It To TL(3)
C
      IF(TEMP .NE. 0.D0) TL(3) = TEMP
      GO TO 770
C
C...Error Message
C
756     CALL BELL
        CALL TITLE(28,13,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'MMMMM.MMM -EG 00002.500 = 2.5 MIN TIME INTERVAL',1)
        GO TO 752
C
C...Erase Old Data And Print A Good Version Of New Data
C
770   CALL CUP(9,63)
      WRITE(IO,772)NULL
      CALL CUP(10,63)
      WRITE(IO,772)NULL
772   FORMAT(A1,'          ')
C
      CALL CUP(9,63)
      WRITE(IO,774) NULL,TL(3)
774   FORMAT(A1,F9.3)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(9,42,'DELTA TIME(MINS)',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Perigee
C
C...Enhance Title
C
840   CALL TITLE(14,6,'ARG OF PERIGEE',1)
C
C...Cursor At Data Line
C
845   CALL CUP(15,8)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (846,848,846,848,850,850,9000,9000), CODE
C
C...Go To Previous Data Item
C
846   CALL TITLE(14,6,'ARG OF PERIGEE',0)
      GO TO 740
C
C...Go To Next Data Item
C
848   CALL TITLE(14,6,'ARG OF PERIGEE',0)
      GO TO 940
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
850   CALL CUP(16,8)
      WRITE(IO,851) NULL,ELM(6)
851   FORMAT(A1,F12.6)
C
852   CALL CUP(15,8)
      WRITE(IO,854)NULL
854   FORMAT(A1,'            ')
C
C...Read New Data Item
C
      CALL CUP(15,8)
```

```
      READ(IN,855,ERR=856) ELM(6)
855   FORMAT(F12.6)
      GO TO 870
C
C...Error Message
C
856     CALL BELL
        CALL TITLE(28,8,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER DEGREE MEASURE IN REAL FORMAT',1)
        GO TO 852
C
C...Erase Old Data And Print A Good Version Of New Data
C
870   CALL CUP(15,8)
      WRITE(IO,872)NULL
      CALL CUP(16,8)
      WRITE(IO,872)NULL
872   FORMAT(A1,'              ')
C
      CALL CUP(15,8)
      WRITE(IO,874) NULL,ELM(6)
874   FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(14,6,'ARG OF PERIGEE',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Rt Ascension
C
C....Enhance Title
C
940   CALL TITLE(14,33,'RT ASCENSION',1)
C
C...Cursor At Data Line
C
945   CALL CUP(15,35)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (946,948,946,948,950,950,9000,9000), CODE
C
C...Go To Previous Data Item
C
946   CALL TITLE(14,33,'RT ASCENSION',0)
      GO TO 840
C
C...Go To Next Data Item
C
948   CALL TITLE(14,33,'RT ASCENSION',0)
      GO TO 1040
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
950   CALL CUP(16,35)
      WRITE(IO,951) NULL,ELM(5)
951   FORMAT(A1,F12.6)
C
952   CALL CUP(15,35)
      WRITE(IO,954)NULL
954   FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(15,35)
      READ(IN,955,ERR=956) ELM(5)
```

```
955   FORMAT(F12.6)
      GO TO 970
C
C...Error Message
C
956     CALL BELL
        CALL TITLE(28,8,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER DEGREE MEASURE IN REAL FORMAT',1)
        GO TO 952
C
C...Erase Old Data And Print A Good Version Of New Data
C
970   CALL CUP(15,35)
      WRITE(IO,972)NULL
      CALL CUP(16,35)
      WRITE(IO,972)NULL
972   FORMAT(A1,'              ')
C
      CALL CUP(15,35)
      WRITE(IO,974) NULL,ELM(5)
974   FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(14,33,'RT ASCENSION',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Inclination
C
C....Enhance Title
C
1040  CALL TITLE(14,59,'INCLINATION',1)
C
C...Cursor At Data Line
C
1045  CALL CUP(15,60)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (1046,1048,1046,1048,1050,1050,9000,9000), CODE
C
C...Go To Previous Data Item
C
1046  CALL TITLE(14,59,'INCLINATION',0)
      GO TO 940
C
C...Go To Next Data Item
C
1048  CALL TITLE(14,59,'INCLINATION',0)
      GO TO 1140
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
1050  CALL CUP(16,60)
      WRITE(IO,1051) NULL,ELM(7)
1051  FORMAT(A1,F12.6)
C
1052  CALL CUP(15,60)
      WRITE(IO,1054)NULL
1054  FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(15,60)
      READ(IN,1055,ERR=1056) ELM(7)
1055  FORMAT(F12.6)
```

```
      GO TO 1070
C
C...Error Message
C
1056    CALL BELL
        CALL TITLE(28,8,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER DEGREE MEASURE IN REAL FORMAT',1)
        GO TO 1052
C
C...Erase Old Data And Print A Good Version Of New Data
C
1070  CALL CUP(15,60)
      WRITE(IO,1072)NULL
      CALL CUP(16,60)
      WRITE(IO,1072)NULL
1072  FORMAT(A1,'              ')
C
      CALL CUP(15,60)
      WRITE(IO,1074) NULL,ELM(7)
1074  FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(14,59,'INCLINATION',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Mean Anomoly
C
C....Enhance Title
C
1140  CALL TITLE(17,6,'MEAN ANOMOLY',1)
C
C...Cursor At Data Line
C
1145  CALL CUP(18,8)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (1146,1148,1146,1148,1150,1150,9000,9000), CODE
C
C...Go To Previous Data Item
C
1146  CALL TITLE(17,6,'MEAN ANOMOLY',0)
      GO TO 1040
C
C...Go To Next Data Item
C
1148  CALL TITLE(17,6,'MEAN ANOMOLY',0)
      GO TO 1240
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
1150  CALL CUP(19,8)
      WRITE(IO,1151) NULL,ELM(1)
1151  FORMAT(A1,F12.6)
C
1152  CALL CUP(18,8)
      WRITE(IO,1154)NULL
1154  FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(18,8)
      READ(IN,1155,ERR=1156) ELM(1)
1155  FORMAT(F12.6)
      GO TO 1170
```

```
C
C...Error Message
C
1156    CALL BELL
        CALL TITLE(28,8,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER DEGREE MEASURE IN REAL FORMAT',1)
        GO TO 1152
C
C...Erase Old Data And Print A Good Version Of New Data
C
1170  CALL CUP(18,8)
      WRITE(IO,1172)NULL
      CALL CUP(19,8)
      WRITE(IO,1172)NULL
1172  FORMAT(A1,'              ')
C
      CALL CUP(18,8)
      WRITE(IO,1174) NULL,ELM(1)
1174  FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(17,6,'MEAN ANOMOLY',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Eccentricity
C
C...Enhance Title
C
1240  CALL TITLE(17,33,'ECCENTRICITY',1)
C
C...Cursor At Data Line
C
1245  CALL CUP(18,35)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (1246,1248,1246,1248,1250,1250,9000,9000), CODE
C
C...Go To Previous Data Item
C
1246  CALL TITLE(17,33,'ECCENTRICITY',0)
      GO TO 1140
C
C...Go To Next Data Item
C
1248  CALL TITLE(17,33,'ECCENTRICITY',0)
      GO TO 1340
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
1250  CALL CUP(19,35)
      WRITE(IO,1251) NULL,ELM(8)
1251  FORMAT(A1,F12.6)
C
1252  CALL CUP(18,35)
      WRITE(IO,1254)NULL
1254  FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(18,35)
      READ(IN,1255,ERR=1256) ELM(8)
1255  FORMAT(F12.6)
      GO TO 1270
C
```

```
C...Error Message
C
1256    CALL BELL
        CALL TITLE(28,3,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER NUMBER IN REAL FORMAT',1)
        GO TO 1252
C
C.......Erase Old Data And Print A Good Version Of New Data
C
1270  CALL CUP(18,35)
      WRITE(IO,1272)NULL
      CALL CUP(19,35)
      WRITE(IO,1272)NULL
1272  FORMAT(A1,'              ')
C
      CALL CUP(18,35)
      WRITE(IO,1274) NULL,ELM(8)
1274  FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(17,33,'ECCENTRICITY',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Mean Motion
C
C....Enhance Title
C
1340  CALL TITLE(17,59,'MEAN MOTION',1)
C
C...CURSOR AT DATA LINE
C
1345  CALL CUP(18,60)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (1346,1348,1346,1348,1350,1350,9000,9000), CODE
C
C...Go To Previous Data Item
C
1346  CALL TITLE(17,59,'MEAN MOTION',0)
      GO TO 1240
C
C...Go To Next Data Item
C
1348  CALL TITLE(17,59,'MEAN MOTION',0)
      GO TO 1440
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
1350  CALL CUP(19,60)
      WRITE(IO,1351) NULL,ELM(2)
1351  FORMAT(A1,F12.6)
C
1352  CALL CUP(18,60)
      WRITE(IO,1354)NULL
1354  FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(18,60)
      READ(IN,1355,ERR=1356) ELM(2)
1355  FORMAT(F12.6)
      GO TO 1370
C
C...Error Message
```

```
C
1356     CALL BELL
         CALL TITLE(28,3,'ERROR IN FORMAT, TRY AGAIN',1)
         CALL TITLE(29,3,'ENTER NUMBER IN REAL FORMAT',1)
         GO TO 1352
C
C...Erase Old Data And Print A Good Version Of New Data
C
1370  CALL CUP(18,60)
      WRITE(IO,1372)NULL
      CALL CUP(19,60)
      WRITE(IO,1372)NULL
1372  FORMAT(A1,'              ')
C
      CALL CUP(18,60)
      WRITE(IO,1374) NULL,ELM(2)
1374  FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(17,59,'MEAN MOTION',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Time Derivative No1
C
C....Enhance Title
C
1440  CALL TITLE(20,54,'TIME DERIVATIVES',1)
      CALL TITLE(21,52,'1ST',1)
C
C...Cursor At Data Line
C
1445  CALL CUP(21,59)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (1446,1448,1446,1448,1450,1450,9000,9000), CODE
C
C...Go To Previous Data Item
C
1446  CALL TITLE(20,54,'TIME DERIVATIVES',0)
      CALL TITLE(21,52,'1ST',0)
      GO TO 1340
C
C...Go To Next Data Item
C
1448  CALL TITLE(20,54,'TIME DERIVATIVES',0)
      CALL TITLE(21,52,'1ST',0)
      GO TO 1540
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
1450  CALL CUP(22,60)
      WRITE(IO,1451) NULL,ELM(3)
1451  FORMAT(A1,F12.6)
C
1452  CALL CUP(21,60)
      WRITE(IO,1454)NULL
1454  FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(21,60)
      READ(IN,1455,ERR=1456) ELM(3)
1455  FORMAT(F12.6)
      GO TO 1470
```

```
C
C...Error Message
C
1456    CALL BELL
        CALL TITLE(28,3,'ERROR IN FORMAT, TRY AGAIN',1)
        CALL TITLE(29,3,'ENTER NUMBER IN REAL FORMAT',1)
        GO TO 1452
C
C...Erase Old Data And Print A Good Version Of New Data
C
1470  CALL CUP(21,60)
      WRITE(IO,1472)NULL
      CALL CUP(22,60)
      WRITE(IO,1472)NULL
1472  FORMAT(A1,'              ')
C
      CALL CUP(21,60)
      WRITE(IO,1474) NULL,ELM(3)
1474  FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(20,54,'TIME DERIVATIVES',0)
      CALL TITLE(21,52,'1ST',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C

C...Entering Of Time Derivative No2
C
C...Enhance Title
C
1540  CALL TITLE(20,54,'TIME DERIVATIVES',1)
      CALL TITLE(23,52,'2ND',1)
C
C...Cursor At Data Line
C
1545  CALL CUP(23,59)
C
C...Check Input For One Of The Function Keys
C
      CALL GETFUN(CODE)
C
C...Go To The Proper Operation
C
      GO TO (1546,1548,1546,1548,1550,1550,9000,9000), CODE
C
C...Go To Previous Data Item
C
1546  CALL TITLE(20,54,'TIME DERIVATIVES',0)
      CALL TITLE(23,52,'2ND',0)
      GO TO 1440
C
C...Go To Next Data Item
C
1548  CALL TITLE(20,54,'TIME DERIVATIVES',0)
      CALL TITLE(23,52,'2ND',0)
      GO TO 40
C
C...Get This Data Item
C....Print Old Item Below Then Erase Input Line
C
1550  CALL CUP(24,60)
      WRITE(IO,1551) NULL,ELM(4)
1551  FORMAT(A1,F12.6)
C
1552  CALL CUP(23,60)
      WRITE(IO,1554)NULL
1554  FORMAT(A1,'              ')
C
C...Read New Data Item
C
      CALL CUP(23,60)
```

```
      READ(IN,1555,ERR=1556) ELM(4)
1555  FORMAT(F12.6)
      GO TO 1570
C
C...Error Message
C
1556            CALL BELL
       CALL TITLE(28,3,'ERROR IN FORMAT, TRY AGAIN',1)
       CALL TITLE(29,3,'ENTER NUMBER IN REAL FORMAT',1)
       GO TO 1552
C
C...Erase Old Data And Print A Good Version Of New Data
C
1570  CALL CUP(23,60)
      WRITE(IO,1572)NULL
      CALL CUP(24,60)
      WRITE(IO,1572)NULL
1572  FORMAT(A1,'              ')
C
      CALL CUP(23,60)
      WRITE(IO,1574) NULL,ELM(4)
1574  FORMAT(A1,F12.6)
C
C...Unenhance Titles And Go On To Next Data Item
C
      CALL TITLE(20,54,'TIME DERIVATIVES',0)
      CALL TITLE(23,52,'2ND',0)
C
C...Clear ''Dialog'' Portion
C
      CALL CLEAR
C
C...Next Item Is The First Item
C
      GO TO 40

C
C...Exit From Editor And Calculate Initial Simulation Data
C
C
C...Reset Scrolling Margins And Prepare To Write Messages At The Bottom
C....Of The Screen
C
9000  CALL TEKSTBM(1,32)
      CALL SELECT_CODE(0)
      CALL DIALOG_LINES(32)
      CALL SELECT_CODE(1)
        CALL CUP(28,1)
C
C...Calculate the initial run time elements
C
      EDT=TM(1)-TM(2)
      TL(1)=TM(1)
      IFG=0
      CALL SGP(TM,ELM,POSN,IFG)
C
C...Calculate Hour Angle
C
        CALL HOURANGLE(TM(1),SGPDAT(7))
C
C...Display Initial Simulation Data In Tek Mode
C
      CALL SELECT_CODE(0)
      CALL SGPDSPT(0)
      CALL DIALOG_LINES(2)
      WRITE(IO,9010)
9010  FORMAT('$','CHECK INPUT? (Y)=YES )')
      READ(IN,9020) IC1
9020    FORMAT(A1)
      IF(IC1.EQ.'Y' .OR. IC1.EQ.'y') THEN
        CALL PAGE
        CALL DIALOG_LINES(32)
        CALL CLEAR_DIALOG_AREA
          CALL SELECT_CODE(1)
          CALL SGPINPUT
```

```
        GO TO 10
      ENDIF
        RETURN
      END

C
C*************************************************************************
C
C   Subroutine   T I T L E
C
C
C     Date                 Author
C  October 9,1985        Mark Lovell
C
C*************************************************************************
C
C   Description:
C     This routine enhances titles or makes them normal
C
C   Calling  Sequence:
C      Call Title(Row,Column,Heading,Code)
C          Row  : Row Number Of The Title
C       Column : Column Number Of The Title
C      Heading : Title To Be Enhansed Or Normalized
C           Code : 1 = Enhance Title
C                  0 = Normalize Title
C
C*************************************************************************
C
C   Declarations:
C
C*************************************************************************
C
       SUBROUTINE TITLE(ROW,COLUMN,HEADING,CODE)
        IMPLICIT COMPLEX (A-Z)
      INTEGER ROW,COLUMN,CODE,IO/6/
      CHARACTER*(*) HEADING
      CHARACTER*1 NULL/'0'X/
C
C*************************************************************************
C
C   Executable Code
C
C*************************************************************************
C
      CALL CUP(ROW,COLUMN)
      IF (CODE .EQ. 0) THEN
        CALL SGR(0)
      ELSE
        CALL SGR(7)
      ENDIF

      WRITE(IO,35)NULL,HEADING
35    FORMAT(A1,' ',A((LEN(HEADING))))
      CALL SGR(0)

      RETURN
      END

C
C*************************************************************************
C
C   Subroutine   G E T F U N
C
C
C     Date                 Author
C  October 9,1985        Mark Lovell
C
C*************************************************************************
C
C   Description:
C     This routine gets one character from the keyboard
```

```
C     Note: no return is needed for this routine
C
C   Calling  Sequence:
C      Call Getfun(Code)
C            Code : Interger To Tell Which Function Key Has Been
C                   Pressed.  Range 1-8 . Ie If Code = 1 Then
C                   F1 Has Been Pressed
C
C*****************************************************************  *****
C
C   Declarations:
C
C*************************************************************************
C
       SUBROUTINE GETFUN(CODE)

      INCLUDE '($IODEF)'
      INTEGER IDS,STORE
      INTEGER TEST,CODE,INCHANNEL
      BYTE BUFFER(4)
      BYTE IOSB(4)
      INTEGER*4 SYS$ASSIGN,SYS$QIOW,SYS$DASSGN
      COMMON/CHANNEL/STORE
C
C*************************************************************************
C
C   Executable Code
C
C*************************************************************************


      TEST = 0
      CODE = 0
C
C...Open A Channel To Screen. Channels Are Not Understood Very Well.
C....This Structure Opens A Channel Once At The Beginning Of The
C....Program Then Retains That Value Regardless Of Later Processes.
C....(The Channel Number Can Be Effected By Operations To Disks,
C....Printers Etc.)
C
      IF(STORE .EQ. 0) THEN
         IDS = SYS$ASSIGN('SYS$COMMAND',STORE,,)
      ENDIF
      INCHANNEL = STORE
C
C...Request Read Of One Character. This Is Placed In BUFFER(1)
C
10    IDS = SYS$QIOW(%VAL(1),%VAL(INCHANNEL),
     +       %VAL(IO$_TTYREADPALL .OR. IO$M_ESCAPE .OR.
     +       IO$M_NOECHO),IOSB,,,BUFFER,%VAL(1),,,,)
C
C...Check For An Input Of The 'ESC' Code
C
      IF (BUFFER(1) .EQ. "33) GO TO 10
C
C...Check For An Input Of The '[' The Second Part Of The
C...Control Sequence
C
      IF (BUFFER(1) .EQ. '[') THEN
        TEST = 1
        GO TO 10
      ENDIF
C
C...Check For An Input Of An 'O'. Signifies That F5-F8 Has Been
C...Pressed
C
      IF (BUFFER(1) .EQ. 'O') THEN
        TEST = 2
        GO TO 10
      ENDIF
C
C...Test The Character That Has Been Received And Set 'CODE'
C
      IF (BUFFER(1) .EQ.'A' .AND. TEST .EQ. 1) CODE = 1
      IF (BUFFER(1) .EQ.'B' .AND. TEST .EQ. 1) CODE = 2
```

```
      IF (BUFFER(1) .EQ.'D' .AND. TEST .EQ. 1) CODE = 3
      IF (BUFFER(1) .EQ.'C' .AND. TEST .EQ. 1) CODE = 4
      IF (BUFFER(1) .EQ.'P' .AND. TEST .EQ. 2) CODE = 5
      IF (BUFFER(1) .EQ.'Q' .AND. TEST .EQ. 2) CODE = 6
      IF (BUFFER(1) .EQ.'R' .AND. TEST .EQ. 2) CODE = 7
      IF (BUFFER(1) .EQ.'S' .AND. TEST .EQ. 2) CODE = 8
C
C...Test To See If A Code Has Been Defined. If So Return. If Not, Go
C....Back For Another Entry
C
      IF (CODE .GT. 0) THEN
C
C...Close The Channel To The Screen
C
         IDS = SYS$DASSGN(INCHANNEL)
           RETURN
      ENDIF
      TEST = 0
      GO TO 10
C
      END
```

```
C
C*****************************************************************
C
C  Subroutine    M O D J U L
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Calculates Modified Julian Date.
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE MODJUL(IDY,MNTH,IYR,ZULU,DATE,IDIR)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION IDAYS(12)
      DATA IDAYS/0,31,59,90,120,151,181,212,243,273,304,334/
C
C...Executable Code
C
      IF(IDIR.EQ.1) GO TO 15
C
C...Compute Number Of Days In Previous Months.
C
      MDAYS=IDAYS(MNTH)
C
C...Compute The Fractional Day Date From Zulu Time
C
      IZULU=ZULU
      SEC=(ZULU-IZULU)*60.
      IHRS=ZULU/100.
      MIN=ZULU-IHRS*100.
      DY=IDY+(IHRS/24.0)+(MIN/1440.0)+(SEC/86400.0)
C
C...Compute # Of Additional Days Due To Leap Year
C
      TEMP=(IYR-48)/4.
      NINT=TEMP
      TEMP=TEMP-NINT
      IF(TEMP.NE.(0.D0)) GO TO 10
      NINT=NINT-1
      IF(MNTH.LE.2) GO TO 10
      MDAYS=MDAYS+1
C
C...Calculate Date
C
10    DATE=33282.0+(365*(IYR-50)+NINT+(MDAYS+DY-1))
      RETURN
C
C...Get Year (Start, 1950)
C
15    IYR=49
      DAT=DATE-33282.
25    IYR=IYR+1
      DAT=DAT-365.
      TEMP=(IYR-1)/4.
      NINT=TEMP
      TEMP=TEMP-NINT
      IF(TEMP.EQ.(0.D0)) DAT=DAT-1.
      IF(DAT.GT.(0.D0)) GO TO 25
      TEMP=(IYR-48)/4.
      NINT=TEMP
      IFG1=1
      IF(TEMP.NE.(0.D0)) GO TO 30
      IFG1=0
      NINT=NINT-1
30    DAT=1.+DATE-(33282.+(365.*(IYR-50))+NINT)
C
C...Get The Month
C
```

```
      DO 35 I=1,11
      SUM=IDAYS(I+1)
      IF((I.GT.1).AND.(IFG1.EQ.0)) SUM=SUM+1.
      IF(DAT.LE.SUM) GO TO 40
35    CONTINUE
40    MNTH=I
      DAT=DAT-IDAYS(I)
      IF((MNTH.GT.2).AND.(IFG1.EQ.0)) DAT=DAT-1
      IDY=DAT
      DAT=DAT-IDY
      HRS=DAT*24.
      IHRS=HRS
      DMINS=(HRS-IHRS)*60.
50    IF(DMINS.LT.(60.D0)) GO TO 45
      DMINS=DMINS-60.D0
      IHRS=IHRS+1
      GO TO 50
45    ZULU=(IHRS*100.D0)+DMINS
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C   Subroutine   S G P
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Calculates Satellite Position
C
C*****************************************************************
C
C...Program Based on Analytics from N/A Pamphlet 55-47
C       5 March 1977, HQ NORAD, a General Perturbation
C       Theory SGP Developed by Aeronutronic-Ford
C       using kozai's theory
C
C...Inputs
C       a) TM - time in MOD JULIAN
C          TM(1) - current time
C          TM(2) - epoch time
C       b) ELM - orbital element vector
C          ELM(1) - epoch mean anomaly, degrees
C              (2) - mean motion, revs/days
C              (3) - first time derivative of ELM(2)
C              (4) - second time derivative of ELM(2)
C              (5) - epoch right ascension, degrees
C              (6) - epoch argument of perigee, degrees
C              (7) - epoch inclination, degrees
C              (8) - epoch eccentricity
C
C...OUTPUT
C       a) POSN(1-3) position vector, earth radii, inertial
C          POSN(4-6) velocity vector, earth radii/min, inertial
C          ( POSN(1),POSN(2),POSN(3) ) = (Rx,Ry,Rz)
C          ( POSN(4),POSN(5),POSN(6) ) = (Vx,Vy,Vz)
C
C       b) SGPDAT(1-14) COMMON BLOCK DUMP
C
C...CONTROL
C       a) IFG -Trace flag
C          IFG = 0, Trace off
C
C...Declarations
        SUBROUTINE SGP(TM,ELM,POSN,IFG)
        IMPLICIT REAL*8(A-Z)
        INTEGER I,IFG
        DIMENSION TM(2),ELM(8),ELM1(8),POSN(6)
        COMMON/SGPELM/SGPDAT(14)
C
        EQUIVALENCE (ELM1(1),M0),(ELM1(2),N0),(ELM1(3),DN0)
        EQUIVALENCE (ELM1(4),DDN0),(ELM1(5),OM0),(ELM1(6),W0)
        EQUIVALENCE (ELM1(7),I0),(ELM1(8),E0)
C
C...Executable Code
C
C...Initialize Constants
C
        PI=3.1415926536D0
        PI2=2.D0*PI
        J2=1082.549D-6
        J3=-2.435D-6
        AE=1.D0
        KE=.0743667785D0
        MU=1.D0
C
C...Angle Changes To Radians
C
        DT=(TM(1)-TM(2))*1440.D0
        DO 35 I=1,8
35      ELM1(I)=ELM(I)
```

```
C
      IF(IFG.EQ.0) GO TO 40
      WRITE(IFG,800) TM(1),TM(2),DT
800   FORMAT(1X,10F13.6,/)
      WRITE(IFG,800) (ELM1(I),I=1,8)
C
40    CONTINUE
      MO=PI2*MO/360.DO
      NO=PI2*NO/1440.DO
      DNO=PI2*DNO/(1440.DO)**2
      DDNO=PI2*DDNO/(1440.DO)**3
      OMO=PI2*OMO/360.DO
      WO=PI2*WO/360.DO
      IO=PI2*IO/360.DO
C
C...The Following Are Calculated
C
      A1=((MU*(KE**2))/NO**2)**(1.DO/3.DO)
      LA1=(3.DO*J2*(AE**2)*(3.DO*(DCOS(IO)**2)-1.DO))
      LA1=LA1/(4.DO*(A1**2)*((1.DO-EO**2)**(3.DO/2.DO)))
      AO=A1*(1.DO-(LA1/3.DO+LA1**2+(134.DO*(LA1**3)/81.DO)))
      PO=AO*(1.DO-EO**2)
      QO=AO*(1.DO-EO)
      LO=MO+WO+OMO
      TEMP=J2*NO*((AE/PO)**2)
      DOM=-3.DO*TEMP*DCOS(IO)/2.DO
      DW=3.DO*TEMP*(5.DO*(DCOS(IO)**2)-1.DO)/4.DO
      IF(IFG.NE.0) WRITE(IFG,800) A1,LA1,AO,PO,QO,LO,DOM,DW
C
C...Update The Mean Orbital Elements (Atmospheric Drag)
C
      A=AO*(NO/(NO+2.DO*DNO*DT+3.DO*DDNO*(DT**2)))**(2.DO/3.DO)
      E=1.D-6
      IF(A.GT.QO) E=1.DO-QO/A
      P=A*(1.DO-E**2)
      IF(IFG.NE.0) WRITE(IFG,800) A,E,P
C
C...Update The Mean Elements For Perturbations(Secular & Long Period)
C
      OMSO=OMO+DOM*DT
      WSO=WO+DW*DT
      AXNSL=E*DCOS(WSO)
      AYNSL=E*DSIN(WSO)-(J3*AE*DSIN(IO))/(2.DO*J2*P)
      LS=LO+(NO+DOM+DW)*DT+DNO*(DT**2)+DDNO*(DT**3)
      L=(J3*AE*AXNSL*DSIN(IO)*(3.DO+5.DO*DCOS(IO)))
      L=LS-L/(4.DO*P*J2*(1.DO+DCOS(IO)))
C
10    IF(L.LT.PI2) GO TO 5
      L=L-PI2
      GO TO 10
C
5     U=L-OMSO
      IF(IFG.NE.0) WRITE(IFG,800) OMSO,WSO,AXNSL,AYNSL,LS,L,U
C
C...Solve Kepler Equation For E+W
C
      EW=U
20    DEW=U+(-AYNSL*DCOS(EW))+AXNSL*DSIN(EW)-EW
      DEW=DEW/(1.DO-(AYNSL*DSIN(EW)+AXNSL*DCOS(EW)))
      IF(DABS(DEW).LT.(1.D-8)) GO TO 15
      EW=EW+DEW
      GO TO 20
C
15    EW=EW+DEW
C
C...Compute Following Preliminary Quantities
C
      ECE=AXNSL*DCOS(EW)+AYNSL*DSIN(EW)
      ESE=AXNSL*DSIN(EW)-AYNSL*DCOS(EW)
      EL2=(AXNSL**2)+(AYNSL**2)
      PL=A*(1.DO-EL2)
      R=A*(1.DO-ECE)
      DR=DSQRT(A*MU*(KE**2))*ESE/R
      RDV=DSQRT(PL*MU*(KE**2))/R
      TEMP=ESE/(1.DO+DSQRT(1.-EL2))
```

```
      SU=A*(DSIN(EW)-(AYNSL+AXNSL*TEMP))/R
      CU=A*(DCOS(EW)+(-AXNSL)+AYNSL*TEMP)/R
      U=DATAN2(SU,CU)
      IF(U.LT.(0.D0)) U=PI2+U
C
      TEMP=(J2*AE*AE)/(PL*PL)
      RK=R+TEMP*PL*(DSIN(I0)**2)*DCOS(2.D0*U)/4.D0
      UK=U-TEMP*(7.D0*(DCOS(I0)**2)-1.D0)*DSIN(2.D0*U)/8.D0
      OMK=OMS0+(3.D0*TEMP*DCOS(I0)*DSIN(2.D0*U))/4.D0
C
30    IF(OMK.LT.PI2) GO TO 25
      OMK=OMK-PI2
      GO TO 30
C
25    IK=I0+(3.D0*TEMP*DSIN(I0)*DCOS(I0)*DCOS(2.D0*U))/4.D0
C
C...Orientation Vectors (M In Orbit Orthagonal To N, N Along Node)
C
      MX=-DSIN(OMK)*DCOS(IK)
      MY=DCOS(OMK)*DCOS(IK)
      MZ=DSIN(IK)
      NX=DCOS(OMK)
      NY=DSIN(OMK)
      NZ=0.D0
      IF(IFG.NE.0) WRITE(IFG,800) MX,MY,MZ,NX,NY,NZ
C
C...Direction Vectors (U In Orbit Plane Along Radius, V
C....Orthagonal To U
C
      UX=MX*DSIN(UK)+NX*DCOS(UK)
      UY=MY*DSIN(UK)+NY*DCOS(UK)
      UZ=MZ*DSIN(UK)+NZ*DCOS(UK)
C
      VX=MX*DCOS(UK)-NX*DSIN(UK)
      VY=MY*DCOS(UK)-NY*DSIN(UK)
      VZ=MZ*DCOS(UK)-NZ*DSIN(UK)
      IF(IFG.NE.0) WRITE(IFG,800) UX,UY,UZ,VX,VY,VZ
C
C...Position And Velocity Vectors
C
      RX=RK*UX
      RY=RK*UY
      RZ=RK*UZ
      DRX=DR*UX+RDV*VX
      DRY=DR*UY+RDV*VY
      DRZ=DR*UZ+RDV*VZ
C
      POSN(1)=RX
      POSN(2)=RY
      POSN(3)=RZ
      POSN(4)=DRX
      POSN(5)=DRY
      POSN(6)=DRZ
C
C...Dump Current Parameters To The Common Block
C
      SGPDAT(1)=WS0
      SGPDAT(2)=OMK
      SGPDAT(3)=IK
      SGPDAT(4)=A
      SGPDAT(5)=E
      SGPDAT(6)=(N0+2.D0*DN0*DT+3.D0*DDN0*(DT**2))*(1440.D0/PI2)
      SGPDAT(8)=RX
      SGPDAT(9)=RY
      SGPDAT(10)=RZ
      SGPDAT(11)=RK
      SGPDAT(12)=DSQRT(DR**2+RDV**2)
      SGPDAT(13)=DR
      SGPDAT(14)=RDV
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C   Subroutine   S G P D S P
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Displays Current Element Set
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPDSP(IOT)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL T1,T2,T3,T4,RMON(12)
      LOGICAL*1 IDASH(78),IC,IFLNAM(14),ERROR
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/SGPELM/SGPDAT(14)
      DATA IDASH/2*' ',76*'-'/
      DATA RMON/'JAN ','FEB ','MAR ','APR ','MAY ','JUN ',
     1 'JUL ','AUG ','SEP ','OCT ','NOV ','DEC '/
C
C...IOT Controls The Write To TI: Or To File
C
      I1=IO
      IF(IOT.EQ.0) GO TO 5
      I1=IOT
      GO TO 10
C
C...Clear the dialog area
C
5       CALL CLEAR_DIALOG_AREA
C
C...Turn Off The Dialog Area
C
      CALL ENABLE_DIALOG_AREA(0)
C
C...Move the graphics cursor to the home position
C
        CALL MOVE(0,3071,IRETURN_CODE)
C
10    WRITE(I1,510) ICON(1)
510   FORMAT(17X,'DATE',7X,'TIME',8X,'RUN NO ',I3)
      IDIR=1
      IF(TM(2).NE.(0.D0)) GO TO 25
      IDY=0
      MNTH=1
      IYR=0
      ZULU=0.D0
      GO TO 30
25    CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
      IZULU = ZULU/100
        DZULU = ZULU - (IZULU*100)
30    WRITE(I1,515) IDY,RMON(MNTH),IYR,IZULU,DZULU,EDT
515   FORMAT(/,1X,'EPOCH',9X,I2,1X,A4,I2,2X,I2,1X,F6.3,5X,
     1 'EPOCH DT (DAYS)',6X,F9.3)
C
      IF(TM(1).NE.(0.D0)) GO TO 35
      IDY=0
      MNTH=1
      IYR=0
      ZULU=0.D0
      GO TO 40
35      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
```

```
C
C...Break ZuLU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
      IZULU = ZULU/100
        DZULU = ZULU - (IZULU*100)
C
40    WRITE(I1,520) IDY,RMON(MNTH),IYR,IZULU,DZULU,TL(2)
520   FORMAT(1X,'SIMULATION',4X,I2,1X,A4,I2,2X,I2,1X,F6.3,5X,
      1 'TOTAL RUN TIME (MINS)',F9.3)
      WRITE(I1,525) TL(3)
525   FORMAT(40X,'DELTA TIME(MINS)',5X,F9.3)
C
      WRITE(I1,560) (IDASH(I),I=1,78)
560   FORMAT(78A1)
C
      WRITE(I1,565)
565   FORMAT(32X,'ORBITAL ELEMENTS')
      WRITE(I1,570)
570   FORMAT(8X,'ARG OF PERIGEE',13X,'RT ASCENSION',14X,
      1 'INCLINATION')
      T1=180.D0*SGPDAT(1)/PI
      T2=180.D0*SGPDAT(2)/PI
      T3=180.D0*SGPDAT(3)/PI
      WRITE(I1,575) T1,T2,T3
575   FORMAT(5X,2(F15.6,11X),F15.6)
      WRITE(I1,580)
580   FORMAT(8X,'SEMI MAJOR AXIS',12X,'ECCENTRICITY',14X,
      1 'MEAN MOTION')
      T1=SGPDAT(4)
      T2=SGPDAT(5)
      T3=SGPDAT(6)
      WRITE(I1,575) T1,T2,T3
C
      WRITE(I1,560) (IDASH(I),I=1,78)
      WRITE(I1,585)
585   FORMAT(31X,'INITIAL CONDITIONS')
      WRITE(I1,590)
590   FORMAT(6X,'HOUR ANGLE',13X,'RX',16X,'RY',17X,'RZ')
      T1=180.D0*SGPDAT(7)/PI
      T2=AE*SGPDAT(8)
      T3=AE*SGPDAT(9)
      T4=AE*SGPDAT(10)
      WRITE(I1,600) T1,T2,T3,T4
600   FORMAT(3X,3(F11.3,8X),F11.3)
      WRITE(I1,605)
605   FORMAT(10X,'R',18X,'V',17X,'VR',17X,'VA')
      T1=AE*SGPDAT(11)
      T2=AE*(1000.D0/60.D0)*SGPDAT(12)
      T3=AE*(1000.D0/60.D0)*SGPDAT(13)
      T4=AE*(1000.D0/60.D0)*SGPDAT(14)
      WRITE(I1,600) T1,T2,T3,T4
      WRITE(I1,560) (IDASH(I),I=1,78)
      WRITE(I1,530)
530   FORMAT(1X)
C
C...Enable Dialog Area
C
        CALL ENABLE_DIALOG_AREA(1)
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P T I M
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Retrieves Apogee, Perigee & Equatorial
C        Crossing Times.
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPTIM(IO)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION FD(3,7),R(3,3),TS(2),POSN(6)
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/TIMSGP/L,N(30),TMS(3,30)
      COMMON/SGPELM/SGPDAT(14)
C
C...Executable Code
C
C...Get The Necessary Timing Data
C
      IFG=0
      L=0
C
C...Get Started
C
      DO 5 I=1,3
      READ(IO,500,ERR=10,END=10) ((FD(J,K),J=1,3),K=1,7)
500   FORMAT(3D25.16)
      R(1,I)=FD(2,1)
      R(2,I)=FD(2,4)
5     R(3,I)=FD(3,6)
C
C...Apogee Time Range
C
65    IF((R(2,1).LT.R(2,2)).AND.(R(2,3).LT.R(2,2))) GO TO 15
      GO TO 20
15    L=L+1
      N(L)=1
      TMS(1,L)=R(1,1)
      TMS(2,L)=R(1,3)
C
C...Perigee Time Range
C
20    IF((R(2,1).GT.R(2,2)).AND.(R(2,3).GT.R(2,2))) GO TO 25
      GO TO 30
25    L=L+1
      N(L)=2
      TMS(1,L)=R(1,1)
      TMS(2,L)=R(1,3)
C
C...Equatorial Crossings
C
30    IF((R(3,1).LE.(0.D0)).AND.(R(3,2).GE.(0.D0))) GO TO 35
      IF((R(3,1).GE.(0.D0)).AND.(R(3,2).LE.(0.D0))) GO TO 40
      GO TO 45
35    L=L+1
      N(L)=3
      TMS(1,L)=R(1,1)
      TMS(2,L)=R(1,2)
      GO TO 45
40    L=L+1
      N(L)=4
      TMS(1,L)=R(1,1)
      TMS(2,L)=R(1,2)
```

```
45    IF(IFG.EQ.1) GO TO 50
      IF((L+1).GE.30) GO TO 50
C
      DO 55 I=1,2
      DO 55 J=1,3
55    R(J,I)=R(J,I+1)
C
      READ(IO,500,ERR=60,END=60) ((FD(I,J),I=1,3),J=1,7)
      R(1,3)=FD(2,1)
      R(2,3)=FD(2,4)
      R(3,3)=FD(3,6)
      GO TO 65
C
60    IFG=1
      GO TO 30
C
50    TS(2)=TM(2)
      IFG=0
C
      IF(L.EQ.0) GO TO 95
      DO 70 I=1,L
      GO TO (75,75,80,80),N(I)
C
C...Apogee/Perigee Search - Interval Sectioning
C
75    T1=TMS(1,I)
      T2=TMS(2,I)
115   T3=T1+(T2-T1)/3.D0
      T4=T1+2.D0*(T2-T1)/3.D0
      TS(1)=T3
      CALL SGP(TS,ELM,POSN,IFG)
      F3=SGPDAT(11)
      TS(1)=T4
      CALL SGP(TS,ELM,POSN,IFG)
      F4=SGPDAT(11)
      IF(N(I).EQ.2) GO TO 85
      IF(F3.LE.F4) GO TO 90
      T2=T4
      GO TO 100
90    T1=T3
      GO TO 100
85    IF(F3.GE.F4) GO TO 105
      T1=T3
      GO TO 100
105   T2=T4
100   IF((T2-T1).LT.(1.D0/(1440.D0*60.D0))) GO TO 110
      GO TO 115
C
110   TMS(3,I)=(T1+T2)/2.D0
      GO TO 70
C
C...Equatorial Crossings - Newton Iteration
C
80    T1=TMS(1,I)
      T2=TMS(2,I)
      TS(1)=T1
      CALL SGP(TS,ELM,POSN,IFG)
      F1=SGPDAT(10)
125   TS(1)=T2
      CALL SGP(TS,ELM,POSN,IFG)
      F2=SGPDAT(10)
C
      T3=T2-F2*(T2-T1)/(F2-F1)
      IF(DABS(T3-T2).LT.(1.D0/(1440.D0*60.D0))) GO TO 120
      T1=T2
      F1=F2
      T2=T3
      GO TO 125
C
120   TMS(3,I)=T3
70    CONTINUE
C
C...Display The Times
C
95    IOT=0
```

```
      CALL TIMDSP(IOT,L,N,TMS)
C
10    REWIND IO
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    T I M D S P
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Displays Timing Information.
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE TIMDSP(IOT,L,N,TMS)
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL RMON(12)
      LOGICAL*1 LB1(80),LB2(80),LB3(80),IC,IC1(5),ICM
      INTEGER*4 IT,IT1
      DIMENSION N(30),TMS(3,30),EVENT(4)
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
C
      DATA RMON/'JAN ','FEB ','MAR ','APR ','MAY ','JUN ',
     1 'JUL ','AUG ','SEP ','OCT ','NOV ','DEC '/
      DATA EVENT/'APOGEE  ','PERIGEE ','S/N     ','N/S     '/
      DATA IC1/"40,"33,"110,"33,"112/
C
C...Executable Code
C
      DO 5 I=1,80
      LB1(I)="40
5     LB2(I)="40
      ENCODE(63,500,LB1)
500   FORMAT(17X,'DATE',10X,'TIME',10X,'TIME',9X,'EVENT')
      ENCODE(52,505,LB2)
505   FORMAT(31X,'ZULU',7X,'SIMULATION')
      I1=IO
      IF(IOT.EQ.0) GO TO 35
      I1=IOT
      WRITE(I1,535)
535   FORMAT(1X)
      GO TO 40
C
C...Clear the dialog area
C
35      CALL CLEAR_DIALOG_AREA
C
40    IDIR=1
      LN=1
      L1=1
      IT=TMS(3,LN)
30    IF(LN.GT.L) GO TO 10
      IT1=TMS(3,LN)
      TS=(TMS(3,LN)-TL(1))*1440.D0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TMS(3,LN),IDIR)
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
      IZULU = ZULU/100
        DZULU = ZULU - (IZULU*100)
C
      IF(L1.NE.1) GO TO 15
      WRITE(I1,510) (LB1(I),I=1,79)
510   FORMAT(79A1)
      WRITE(I1,510) (LB2(I),I=1,79)
15    IF((L1.NE.1).AND.(IT.EQ.IT1)) GO TO 20
      IF(IT.NE.IT1) IT=IT1
      ENCODE(66,515,LB3) IDY,RMON(MNTH),IYR,IZULU,DZULU,TS,EVENT(N(LN))
515   FORMAT(15X,I2,1X,A4,I2,5X,I2,1X,F6.3,3X,F9.3,8X,A8)
```

```
      GO TO 25
C
20    ENCODE(66,520,LB3) IZULU,DZULU,TS,EVENT(N(LN))
520   FORMAT(29X,I2,1X,F6.3,3X,F9.3,8X,A8)
25    WRITE(I1,510) (LB3(I),I=1,66)
C
      L1=L1+1
      LN=LN+1
      IF(L1.LE.16) GO TO 30
      IF(I1.NE.10) GO TO 30
      L1=1
      WRITE(I1,525)
525   FORMAT('$',' OUTPUT MORE? (Y)=YES )')
      READ(IN,530) ICM
530   FORMAT(A1)
      IF(ICM.NE."131) GO TO 10
C
C...Clear the dialog area
C
        CALL CLEAR_DIALOG_AREA
C
      GO TO 30
C
10    RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    R D A T E
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Reads Input Date
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE RDATE(BDAT,NC,IDY,MNTH,IYR,IERR)
C
      LOGICAL*1 BDAT(10),BF1(4),BL
      DIMENSION RMON(12)
C
      EQUIVALENCE (BF1,RBF)
      DATA RMON/'JAN ','FEB ','MAR ','APR ','MAY ','JUN ',
     1 'JUL ','AUG ','SEP ','OCT ','NOV ','DEC '/,BL/' '/
C
C...Executable Code
C
      IERR=0
      IF(NC.EQ.0) GO TO 10
      BF1(1)=BDAT(NC-1)
      BF1(2)=BDAT(NC)
      DECODE(2,500,BF1,ERR=10) IYR
500   FORMAT(I2)
      DO 15 J=1,4
15    BF1(J)=BL
      J1=NC-5
      J2=NC-3
      K=0
      DO 20 J=J1,J2
      K=K+1
20    BF1(K)=BDAT(J)
      DO 25 J=1,12
      IF(RBF.EQ.RMON(J)) GO TO 30
25    CONTINUE
C
C...Error In Format
C
10    IERR=1
      RETURN
C
30    MNTH=J
      IF(NC.EQ.8) GO TO 35
      DECODE(2,500,BDAT(1),ERR=10) IDY
      GO TO 5
35    DECODE(1,505,BDAT(1),ERR=10) IDY
505   FORMAT(I1)
C
5     RETURN
C
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P R U N
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Runs The Simulation
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPRUN
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 FLNAME(26),IC
      DIMENSION FD(3,7),POSN(6)
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/SGPELM/SGPDAT(14)
C
C...Executable Code
C
      T=0.D0
      IFG=0
      ENCODE(14,500,FLNAME) ICON(1)
500   FORMAT('SGPSIM.DAT;',I3)
        OPEN(UNIT=17,FILE=FLNAME,STATUS='UNKNOWN',IOSTAT=IOS)
C
25    CALL SGP(TM,ELM,POSN,IFG)
      FD(1,1)=TM(2)
      FD(2,1)=TM(1)
      FD(3,1)=T
      DO 5 I=1,3
      FD(I,2)=SGPDAT(I)
      FD(I,3)=SGPDAT(I+3)
      FD(I,6)=POSN(I)*AE
5     FD(I,7)=POSN(I+3)*(AE*1000.D0)/60.D0
C
      FD(1,4)=SGPDAT(7)+T*(.7292115146D-4)*60.D0
15    IF(FD(1,4).LT.(2.*PI)) GO TO 10
      FD(1,4)=FD(1,4)-2.*PI
      GO TO 15
C
10    FD(2,4)=SGPDAT(11)*AE
      FD(3,4)=SGPDAT(12)*(AE*1000.D0)/60.D0
      FD(1,5)=SGPDAT(13)*(AE*1000.D0)/60.D0
      FD(2,5)=SGPDAT(14)*(AE*1000.D0)/60.D0
      FD(3,5)=0.D0
C
      WRITE(17,505) ((FD(I,J),I=1,3),J=1,7)
505   FORMAT(3D25.16)
      T=T+TL(3)
      TM(1)=TM(1)+TL(3)/1440.D0
      IF(T.LE.TL(2)) GO TO 25
C
      REWIND 17
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P H D R
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Outputs Listing Headers
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPHDR(IO,IOT)
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 LB1(80),IFLNAM(26),IC
      DIMENSION IOT(6),POSN(6)
      COMMON/GENDAT/IC(8),IN,I1,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/TIMSGP/L,N(30),TMS(3,30)
C
C...Executable Code
C
C...Get Current Element Set - At Start Of Simulation
C
      TM(1)=TL(1)
      IFG=0
      CALL SGP(TM,ELM,POSN,IFG)
      ENCODE(14,500,IFLNAM) ICON(1)
500   FORMAT('SGPLST.SUM;',I3)
      IO1=1
        OPEN(UNIT=IO1,FILE=IFLNAM,STATUS='UNKNOWN')
      CALL SGPDSP(IO1)
      IF(IOT(1).EQ.0) GO TO 5
C
      CALL TIMDSP(IO1,L,N,TMS)
5     REWIND IO1
15    READ(IO1,505,ERR=10,END=10) (LB1(I),I=1,80)
505   FORMAT(80A1)
      WRITE(IO,510) (LB1(I),I=1,80)
510   FORMAT(26X,80A1)
      GO TO 15
C
10    CALL CLOSE(IO1)
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P F M T
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Builds Listing Formatted Files
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPFMT(I1,IOT,TREL)
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 IC
      DIMENSION FD(3,7),IOT(6),D(9),XE(3),VE(3),XI(3),XO(3),A(3)
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
C
C...Executable Code
C
C
      WZ=(.7292115146D-4)*60.D0
      F=1.D0/298.25D0
C
55    READ(I1,500,ERR=10,END=10) ((FD(I,J),I=1,3),J=1,7)
500   FORMAT(3D25.16)
C
      DO 15 I=1,4
      D(1)=FD(2,1)
      D(2)=FD(3,1)
      D(3)=FD(3,1)-TREL
      IF(IOT(I+1).EQ.0) GO TO 15
      IF(I.NE.1) GO TO 25
C
C...Inertial System
C
20    DO 45 J=1,3
      D(3+J)=FD(J,6)
45    D(6+J)=FD(J,7)
      GO TO 40
C
C...Earth Cartesian System
C
25    HA=FD(1,4)
      DC=DCOS(HA)
      DS=DSIN(HA)
      D(4)=FD(1,6)*DC+FD(2,6)*DS
      D(5)=-(FD(1,6)*DS-FD(2,6)*DC)
      D(6)=FD(3,6)
      D(7)=(FD(1,7)*DC+FD(2,7)*DS)+(WZ/60.D0)*FD(2,6)*1000.D0
      D(8)=FD(2,7)*DC-(FD(1,7)*DS+(WZ/60.D0)*FD(1,6)*1000.D0)
      D(9)=FD(3,7)
      IF(I.EQ.2) GO TO 40
C
C...Geographic - Longitude, Geocentric And Geographic Latitudes
C
      DO 50 J=1,3
      XE(J)=D(3+J)
50    VE(J)=D(6+J)
      D(4)=DATAN2(XE(2),XE(1))
      IF(D(4).LT.(0.D0)) D(4)=2.*PI+D(4)
C
      TMP=DSQRT(XE(1)**2+XE(2)**2)
      D(5)=DATAN2(XE(3),TMP)*180.D0/PI
      TMP=TMP*(1.D0-F)**2.
      D(6)=DATAN2(XE(3),TMP)*180.D0/PI
      D(7)=FD(3,4)
      D(8)=DSQRT(VE(1)**2+VE(2)**2+VE(3)**2)
```

```
      R1=(AE**2)*(1.-F)**2.
      R1=DSQRT(R1/(1.-(2.*F-F**2.)*DCOS(D(5))**2.))
      D(4)=D(4)*180./PI
      D(9)=FD(2,4)-R1
      IF(I.EQ.3) GO TO 40
C
C...Orbital System
C
      DO 60 J=1,3
      A(J)=FD(J,2)
60    XI(J)=FD(J,6)
      IDIR=1
      CALL ECOECI(XI,XO,A,IDIR)
      D(4)=DATAN2(XO(2),XO(1))
      IF(D(4).LT.(0.D0)) D(4)=2.*PI+D(4)
      D(4)=D(4)*180.D0/PI
      D(5)=FD(2,4)
      D(6)=D(9)
      DO 65 J=1,3
65    D(6+J)=FD(J,5)
C
40    WRITE(I,510) (D(J),J=1,9)
510   FORMAT(1X,D25.16,8F12.2)
15    CONTINUE
      GO TO 55
C
10    REWIND I1
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    E C O E C I
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Transforms Vectors Between ECO And ECI Coords
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE ECOECI(XI,XO,ANG,IDIR)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION XI(3),XO(3),ANG(3)
C
C...Executable Code
C
C...IDIR Is A Flag  0 = Orbital To Inertial
C                   1 = Inertial To Orbital
C...ANG: Perigee, Rt Ascension, Inclination
C
      DC1=DCOS(ANG(1))
      DS1=DSIN(ANG(1))
      DC2=DCOS(ANG(2))
      DS2=DSIN(ANG(2))
      DC3=DCOS(ANG(3))
      DS3=DSIN(ANG(3))
C
      PX=DC1*DC2-DS1*DS2*DC3
      PY=DC1*DS2+DS1*DC2*DC3
      PZ=DS1*DS3
      QX=-(DS1*DC2+DC1*DS2*DC3)
      QY=-(DS1*DS2-DC1*DC2*DC3)
      QZ=DC1*DS3
      WX=DS2*DS3
      WY=-DC2*DS3
      WZ=DC3
C
      IF(IDIR.NE.0) GO TO 5
      XI(1)=PX*XO(1)+QX*XO(2)+WX*XO(3)
      XI(2)=PY*XO(1)+QY*XO(2)+WY*XO(3)
      XI(3)=PZ*XO(1)+QZ*XO(2)+WZ*XO(3)
      RETURN
C
5     XO(1)=PX*XI(1)+PY*XI(2)+PZ*XI(3)
      XO(2)=QX*XI(1)+QY*XI(2)+QZ*XI(3)
      XO(3)=WX*XI(1)+WY*XI(2)+WZ*XI(3)
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P L S T
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Outputs SGPSAT Display Files
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPLST(I1,IOT)
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL*1 IFLNAM(14),IOUT(132),L1(132),L2(132),L3(132),BL
      LOGICAL*1 LBF(36)
      REAL RMON(12)
      DIMENSION IOT(6)
      DATA RMON/'JAN ','FEB ','MAR ','APR ','MAY ','JUN ',
     1 'JUL ','AUG ','SEP ','OCT ','NOV ','DEC '/
      DATA BL/' '/
C
C...Executable Code
C
C...Output Loop
C
      IDIR=1
      DO 20 I=1,4
      IFG=1
      IF(IOT(I+1).EQ.0) GO TO 20
      REWIND I
C
10    WRITE(I1,520)
520   FORMAT('1')
C
      CALL SGPLBL(I,L1,L2,L3)
C
35    WRITE(I1,530) (L1(J),J=1,132)
      WRITE(I1,530) (L2(J),J=1,132)
      WRITE(I1,530) (L3(J),J=1,132)
      L=3
      IFG2=1
C
30    READ(I,515,END=25,ERR=25) DATE,T,T1,(IOUT(52+J),J=1,72)
515   FORMAT(1X,D25.16,2F12.2,72A1)
      DO 95 J=1,52
95    IOUT(J)=BL
      CALL MODJUL(IDY,MNTH,IYR,ZULU,DATE,IDIR)
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
      IZULU = ZULU/100
        DZULU = ZULU - (IZULU*100)
C
      IF(IFG.EQ.0) GO TO 55
      IFG=0
      IDY1=IDY
C
55    IF(IDY1.NE.IDY) IFG2=1
      IF(IFG2.EQ.0) GO TO 60
      IF(L.NE.3) IDY1=IDY
      IFG2=0
      ENCODE(9,510,LBF) IDY,RMON(MNTH),IYR
510   FORMAT(I2,1X,A4,I2)
      DO 65 J=1,9
65    IOUT(7+J)=LBF(J)
C
60    N=24
```

```
      IF(IOT(6).EQ.0) GO TO 70
      ENCODE(36,535,LBF) IZULU,DZULU,T,T1
535   FORMAT(' ',3X,I2,1X,F5.2,F12.2,F12.2)
      N=36
      GO TO 75
70    ENCODE(24,540,LBF) IZULU,DZULU,T
540   FORMAT(' ',3X,I2,1X,F5.2,F12.2)
75    DO 80 J=1,N
80    IOUT(16+J)=LBF(J)
C
      WRITE(I1,530) (IOUT(J),J=1,132)
530   FORMAT(132A1)
      L=L+1
      IF(L.LT.57) GO TO 30
      WRITE(I1,520)
      GO TO 35
C
25    WRITE(I1,520)
20    CONTINUE
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P L B L
C
C     Date          Author
C
C   March 1980      Roy McPherson
C
C*****************************************************************
C
C Description:
C      This Subroutine Builds Output Labels
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPLBL(IT,L1,L2,L3)
C
      LOGICAL*1 L1(132),L2(132),L3(132)
C
C...Executable Code
C
      GO TO (5,10,15,20),IT
C
C...Inertial
C
5     ENCODE(132,500,L1)
500   FORMAT(50X,'CARTESIAN COORDINATES - INERTIAL',50X)
      GO TO 25
C
C...Earth Rotating
C
10    ENCODE(132,505,L1)
505   FORMAT(46X,'CARTESIAN COORDINATES - EARTH(ROTATING)',47X)
25    ENCODE(132,510,L2)
510   FORMAT(10X,'DATE',8X,'TIME',9X,'T',9X,'T REL',9X,'X'
     1 ,11X,'Y',11X,'Z',10X,'VX',10X,'VY',10X,'VZ',11X)
      ENCODE(132,515,L3)
515   FORMAT(22X,'ZULU',7X,'(MIN)',7X,'(MIN)',7X,'(KMS)',7X,
     1 '(KMS)',7X,'(KMS)',6X,'(M/SEC)',5X,'(M/SEC)',5X,'(M/SEC)',
     2 9X)
      RETURN
C
C...Geographic
C
15    ENCODE(132,520,L1)
520   FORMAT(55X,'GEOGRAPHIC COORDINATES',55X)
      ENCODE(132,525,L2)
525   FORMAT(10X,'DATE',8X,'TIME',9X,'T',9X,'T REL',5X,'LONGITUDE',
     1 5X,'LAT(C)',6X,'LAT(G)',7X,'V(I)',8X,'V(E)',6X,
     2 'ALTITUDE',8X)
      ENCODE(132,530,L3)
530   FORMAT(22X,'ZULU',7X,'(MIN)',7X,'(MIN)',5X,'(DEG EAST)',4X,
     1 '(DEG)',7X,'(DEG)',7X,'(M/SEC)',5X,'(M/SEC)',5X,'(KMS)',10X)
      RETURN
C
C...Orbital
C
20    ENCODE(132,535,L1)
535   FORMAT(56X,'ORBITAL COORDINATES',57X)
      ENCODE(132,540,L2)
540   FORMAT(10X,'DATE',8X,'TIME',9X,'T',9X,'T REL',4X,'TRUE ANOMALY'
     1 ,5X,'R',8X,'ALTITUDE',6X,'V(R)',8X,'V(A)',8X,'V(N)',10X)
      ENCODE(132,545,L3)
545   FORMAT(22X,'ZULU',7X,'(MIN)',7X,'(MIN)',7X,'(DEG)',7X,
     1 '(KMS)',7X,'(KMS)',7X,'(M/SEC)',5X,'(M/SEC)',5X,
     2 '(M/SEC)',8X)
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    S G P R E D F I L E
C
C     Date          Author
C
C   Oct 1985       Mark W. Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Reads A Set Of Epoch Conditions From
C        A File Called Epochdata.
C
C*****************************************************************
C
C...Declarations
C
        SUBROUTINE SGPREDFILE(ERROR)
C
         IMPLICIT REAL*8(A-H,O-Z)
       REAL*8 POSN(6),HA,HAO,ZULU,DT1,WLON,ELON
       REAL*8 ELM,TM,EDT,TL,SGPDAT
              REAL*8 PI,AE
       REAL*8 IERR
       LOGICAL*1 CODE,IC,IC1,BDAT(10),ERROR
       CHARACTER*1 DUMMY
       INTEGER I,UNITNO,IFG,IN,IO,ICON
       INTEGER IHA,IZULU
       COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
         COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
       COMMON/SGPELM/SGPDAT(14)
C
C...Executable Code
C....If This Is The First Run, Open The File And Read Past The
C....Instructions To Prepare For Data Input
C
        IF(ICON(1) .EQ. 1) THEN
       UNITNO=13
         OPEN(UNIT=UNITNO,FILE='EPOCHDATA.TXT',STATUS='OLD'.ERR=400)
         REWIND UNITNO
       DO 5 I=1,13
5      READ(UNITNO,8,ERR=300)DUMMY
8      FORMAT(A1)
        ENDIF
C
C...Assume No File Errors To Start
C
       ERROR = 0
C
C
C...Read Orbital Elements
C
         DO 10 I=1,8
10       READ(UNITNO,25,END=500,ERR=300) ELM(I)
25       FORMAT (D25.16)
C
C...Read Epoch Time And Start Time
C
C...Epoch Date Can Be Read In Modified Julian Or In Standard Format
C....S=Standard Format  J=Julian Format
C
       READ(UNITNO,45,ERR=300) IC1
       IF(IC1.EQ.'J') GO TO 27
C
C....If S: Then Enter Date And Zulu Time In The Form
C....DD MMM YR HHHH.HHH   EQ.  Enter-22 APR 65 2359.99 Into File
C
         NC=9
       READ(UNITNO,26,ERR=300) (BDAT(I),I=1,NC),ZULU
26     FORMAT(9A1,1X,D25.16)
       CALL RDATE(BDAT,NC,IDY,MNTH,IYR,IERR)
       IF(IERR.GT.0) GO TO 300
C
```

```
C...Calculate Julian Date
C
      IDIR=0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
      GO TO 30
C
C...Otherwise Read Julian Date
C
27    READ(UNITNO,25,ERR=300) TM(2)
C
C...Simulation Date Can Be Read In Modified Julian
C...Or In Standard  Format. S=Standard;J=Julian
C
30    READ(UNITNO,45,ERR=300) IC1
      IF(IC1.EQ.'J') GO TO 35
C
C...If S; Then Date And Zulu Time Read In The Form
C...DD MMM YR HHMM.MMM   EG.  Enter-22 APR 65 2359.99 Into The File
C
      NC=9
      READ(UNITNO,26,ERR=300) (BDAT(I),I=1,NC),ZULU
      CALL RDATE(BDAT,NC,IDY,MNTH,IYR,IERR)
      IF(IERR.GT.0) GO TO 300
C
C...Convert To Julian Time
C
      IDIR=0
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
      GO TO 37
C
C
C...Otherwise Read In The Julian Simulation Date
C
35    READ(UNITNO,25,ERR=300) TM(1)
      IDIR=1
      CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
C
C
37    EDT=TM(1)-TM(2)
      TL(1)=TM(1)
      IFG=0
      CALL SGP(TM,ELM,POSN,IFG)
C
C
C...Read Total Time And Delta T - Min.
C
        DO 40 I =2,3
40      READ (UNITNO,25,ERR=300) TL(I)

C
C...Format For Reading One Character
C
45      FORMAT(A1)
C
C...Calculate Hour Angle
C
        CALL HOURANGLE(TM(1),SGPDAT(7))
C
C...Display The Information On The Screen
C....Select Tek Mode
C
200     CALL SELECT_CODE(0)
      CALL SGPDSP(0)
C
C....Restrict Dialog Lines
C
            CALL DIALOG_LINES(4)
220   RETURN
C
C...Error In The File Or Its Data
C
300   WRITE(IO,310)
310   FORMAT(' *ERROR* IN READING DATA. FIX THE FILE DATA OR ENTER')
      WRITE(IO,320)
320   FORMAT('     THE DATA ON THE SCREEN.')
```

```
      ERROR = 1
      GO TO 220
C
C
400   WRITE(IO,410)
410   FORMAT(' *ERROR* IN OPENING "EPOCHDATA.TXT". FIX THE FILE OR')
      WRITE(IO,420)
420   FORMAT('      ENTER THE DATA ON THE SCREEN.')
      ERROR = 1
      GO TO 220
C
C
500   WRITE(IO,510)
510   FORMAT(' *NO MORE DATA* IN "EPOCHDATA.TXT". FIX THE FILE OR')
      WRITE(IO,520)
520   FORMAT('      ENTER THE DATA ON THE SCREEN.')
      ERROR = 1
      GO TO 220
      END
```

```
# include math
/*-------------------------------------------------------------------------
 *  Routine    C A L C U L A T E   H O U R   A N G L E
 *
 *
 *
 *    Date:              Author:
 *  September 24, 1985  Mark W. Lovell
 *
 *-------------------------------------------------------------------------
 *
 *    Description:
 *      This routine calculates Greenwich hour angle in radians, given time
 *        in Modified Julian format. Formulas used can be found in
 *        Section 7.4 in Orbit Determination and Prediction Mathematics
 *
 *    Calling Sequence:
 *
 *     Hourangle( Time, Hour_angle )
 *     Time       : Time (in Modified Julian) for which to calculate hour angle
 *     Hour_angle : Hour angle at the given time
 *
 *-------------------------------------------------------------------------
 *
 */

Hourangle( time, hour_angle )
 double *time,*hour_angle;

 {
  double sun_epsilon();
  double delta_epsilon();     /* routines from 'solarvariables.c' module */
  double delta_psi();

  double agm;                 /* longitude of greenwich */
  double dw,de;               /* See Section 7.4 in Orbit Determination
                                    and Prediction Mathematics     */
  double e;                   /* See Section 7.2 in same document*/
  double dtr = 3.1415926536/180;  /* degree to radian convertion */
  double temp;                 /* temporary storage */

  double int_store,*for_modf;  /* needed for modf function but not used */
   for_modf = &int_store;            /* for anything in the program */

 /* Sovle for 'e' ( mean obliquity - degrees ) as in Section 7.2 */

    e = sun_epsilon(*time);

 /* Solve for 'dw' ( nutation in longitude - degrees) and
    'de' ( nutation in obliquity - degrees) as in Section 7.5 */

    dw = delta_psi(*time);
    de = delta_epsilon(*time);

 /* Now solve for 'agm' and 'dw*cos(e + de)' as in Section 7.4 */

    agm = 100.075542 + 360.985647346*(*time-33282);
    agm = agm + 0.29e-12*(*time-33282)*(*time-33282);

    e = e*dtr;
    de = de*dtr;
    temp = dw*cos(e + de);


 /* Finally solve for hour_angle as in Section 7.4 */

   *hour_angle = agm + temp;

      /* Calculate hour_angle mod 360 - (degrees) */

      *hour_angle = *hour_angle / 360;
      *hour_angle = modf(*hour_angle,for_modf);
      *hour_angle = *hour_angle * 360;
```

```
    /* Convert hour_angle to radians */
     *hour_angle = *hour_angle * dtr;
}
```

```
#include stdio
#include descrip
#include ssdef
/*------------------------------------------------------------------------
 *  Routine    E R A S E   I N   D I S P L A Y
 *
 *
 *
 *    Date:                Author:
 *   October 4, 1985    Mark W. Lovell
 *
 *-----------------------------------------------------------------------
 *
 *    Description:
 *      This routine erases all or part of the display
 *
 *    Calling Sequence:
 *
 *       ED( Erase_mode,error )
 *      Erase_mode      : Erase extent. Range 2,1 or 0
 *      Error           : 1 = error,Enable_mode no 1 or 0
 *                        0 = no error
 *-----------------------------------------------------------------------
 *
 */
ED( erase_mode,error )
 int *erase_mode,*error;
{
 int esc = 0x1B;

  *error = 0;

/* Check that erase_mode is 0,1 or 2,if not set error flag */

if ( *erase_mode )2 || *erase_mode (0  )
  {
  *error = 1;
  printf("Error in ED - erase_mode %d must = 2,1 or 0 \n",*erase_mode);
}

else
  {
   /* if there was no error, call the proper escape sequence */

   printf(" %c[%dJ\n",esc,*erase_mode);
  }

  }

/*------------------------------------------------------------------------
 *  Routine    R E S E T   O R I G I N   M O D E
 *
 *
 *
 *    Date:                Author:
 *   October 4, 1985    Mark W. Lovell
 *
 *-----------------------------------------------------------------------
 *
 *    Description:
 *      This routine puts to origin mode to 'absolute'
 *
 *    Calling Sequence:
 *
 *       Absolute()
 *-----------------------------------------------------------------------
 *
 */
Absolute()
{
 int esc = 0x1B;

  printf(" %c[?6l\n",esc);

  }
```

```
/*-------------------------------------------------------------------------
 *  Routine    C U R S O R   P O S I T I O N
 *
 *
 *
 *    Date:              Author:
 *   October 4, 1985    Mark W. Lovell
 *
 *-------------------------------------------------------------------------
 *
 *    Description:
 *      This routine sets the row and column number of the cursor
 *
 *    Calling Sequence:
 *
 *       CUP( Row,Column)
 *      Row              : Row number of cursor
 *      Column           : Column number of cursor
 *-------------------------------------------------------------------------
 *
 */
Cup( row, column )
 int *row,*column;
{
 int esc = 0x1B;

  printf(" %c[%d;%dH",esc,*row,*column);
 }

/*-------------------------------------------------------------------------
 *  Routine    E D I T   M A R G I N S
 *
 *
 *
 *    Date:              Author:
 *   October 4, 1985    Mark W. Lovell
 *
 *-------------------------------------------------------------------------
 *
 *    Description:
 *      This routine sets the edit margins
 *
 *    Calling Sequence:
 *
 *       Tekstbm( Top_margin,Bottom_margin )
 *      Top_margin      : Row number of top margin
 *      Bottom_margin   : Row number of bottom margin
 *-------------------------------------------------------------------------
 *
 */
Tekstbm( top_margin, bottom_margin )
 int *top_margin,*bottom_margin;
{
 int esc = 0x1B;

  printf(" %c[%d;%dr",esc,*top_margin,*bottom_margin);
 }

/*-------------------------------------------------------------------------
 *  Routine    G R A P H I C   R E N D I T I O N
 *
 *
 *
 *    Date:              Author:
 *   October 7, 1985    Mark W. Lovell
 *
 *-------------------------------------------------------------------------
 *
 *    Description:
 *      This routine sets the style of text output
 *
 *    Calling Sequence:
 *
 *       SGR( code )
```

```
*       Code          : Code of text rendition
*-------------------------------------------------------------------------
*
*/
Sgr( code )
 int *code;
{
 int esc = 0x1B;

  printf(" %c[%dm",esc,*code);
 }

/*-------------------------------------------------------------------------
*  Routine    B E L L
*
*
*
*    Date:                Author:
*   October 14, 1985      Mark W. Lovell
*
*-------------------------------------------------------------------------
*
*    Description:
*      This routine sounds bell
*
*    Calling Sequence:
*
*       Bell()
*
*-------------------------------------------------------------------------
*
*/
Bell()
{
 int ring = 0x7;

  printf("%c",ring);
 }
```

```
C
C*****************************************************************
C
C  Subroutine    S G P I N P U T
C
C     Date          Author
C
C   Oct 1985       Mark W. Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Displays The Current Element Set
C         And Receives New Data
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE SGPINPUT
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL T1,T2,T3,T4,RMON(12)
      LOGICAL*1 IDASH(78),IC,IFLNAM(14),ERROR
      CHARACTER NULL/'0'X/
      COMMON/GENDAT/IC(8),IN,IO,ICON(4),PI,AE
      COMMON/ELMSGP/ELM(8),TM(2),EDT,TL(3)
      COMMON/SGPELM/SGPDAT(14)
      DATA IDASH/2*' ',76*'-'/
      DATA RMON/'JAN ','FEB ','MAR ','APR ','MAY ','JUN ',
     1 'JUL ','AUG ','SEP ','OCT ','NOV ','DEC '/
C
C...Executable Code
C
C...Clear the dialog area
C
       CALL ED(2,ERROR)
       IF (ERROR) STOP
C
C...Move the graphics cursor to the home position
C
       CALL ABSOLUTE
       CALL CUP(1,1)
C
C...Write Out Headings
C....Underline Main Title
C
      CALL CUP(1,10)
      CALL SGR(4)
      WRITE(IO,500)NULL
500   FORMAT(A1,' INITIAL CONDITIONS')
C
C...Return To Normal Output Ie. No Underlining
C
      CALL SGR(0)
C
      WRITE(IO,540)
10    WRITE(IO,510)
510   FORMAT(32X,'DATE',7X,'TIME',8X,'JULIAN DATE')
      IDIR=1
C
C...Epoch Date
C
      IF(TM(2).EQ.(0.D0)) THEN
C
C...Reset Standard Time To JAN 1 1950 At 0h
C
         IDY=1
        MNTH=1
        IYR=50
        ZULU=0.D0
        TM(2)=33282.D0
C
       ELSE
C
```

```
C...Calculate Standard Time
C
         CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(2),IDIR)
C
      ENDIF
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
         IZULU = ZULU/100
           DZULU = ZULU - (IZULU*100)
C
      WRITE(IO,515) IDY,RMON(MNTH),IYR,IZULU,DZULU,TM(2)
515   FORMAT(/,15X,'EPOCH',9X,I2,1X,A4,I2,4X,I2,1X,F6.3,3X,
     1 F12.6)
      WRITE (IO,540)
C
C...Simulation Date
C
      IF(TM(1).EQ.(0.D0)) THEN
C
C...Reset Standard Time To JAN 1 1950 At 0h
C
         IDY=1
         MNTH=1
         IYR=50
         ZULU=0.D0
         TM(1)=33282.D0
      ELSE
C
C....Else Calculate Standard Time
C
           CALL MODJUL(IDY,MNTH,IYR,ZULU,TM(1),IDIR)
C
        ENDIF
C
C...Break ZULU From (HHMM.MM) Into IZULU (HH) And DZULU (MM.MM)
C
         IZULU = ZULU/100
           DZULU = ZULU - (IZULU*100)
C
      WRITE(IO,520) IDY,RMON(MNTH),IYR,IZULU,DZULU,TM(1)
520   FORMAT(15X,'SIMULATION',4X,I2,1X,A4,I2,4X,I2,1X,F6.3,3X,
     1 F12.6)
      WRITE (IO,540)
C
C...Run Time And Delta Time
C
      WRITE(IO,525) TL(2),TL(3)
525   FORMAT(8X,'TOTAL RUN(MINS)',3X,F9.3,9X,
     1 'DELTA TIME(MINS)',3X,F9.3)
      WRITE (IO,540)
C
C...Write Out Orbital Elements
C
C
C...Write Out Headings
C....Underline Main Title
C
      CALL CUP(12,30)
      CALL SGR(4)
      WRITE(IO,565)NULL
565   FORMAT(A1,' ORBITAL ELEMENTS')
C
C...Return To Normal Output Ie. No Underlining
C
      CALL SGR(0)
C
      WRITE(IO,540)
      WRITE(IO,570)
570   FORMAT(8X,'ARG OF PERIGEE',13X,'RT ASCENSION',14X,
     1 'INCLINATION')
C
      T1=ELM(6)
      T2=ELM(5)
      T3=ELM(7)
```

```
      WRITE(IO,575) T1,T2,T3
575   FORMAT(8X,2(F12.6,14X),F12.6)
      WRITE (IO,540)
C
      WRITE(IO,580)
580   FORMAT(8X,'MEAN ANOMOLY',15X,'ECCENTRICITY',14X,
     1 'MEAN MOTION')
      T1=ELM(1)
      T2=ELM(8)
      T3=ELM(2)
      WRITE(IO,575) T1,T2,T3
C
      WRITE (IO,540)
540   FORMAT (1X)
C
      WRITE(IO,590)
590   FORMAT(56X,'TIME DERIVATIVES')
      WRITE(IO,600) ELM(3)
600   FORMAT(54X,'1ST ',2X,F12.6)
      WRITE (IO,540)
      WRITE(IO,610) ELM(4)
610   FORMAT(54X,'2ND ',2X,F12.6)
C     WRITE (IO,540)
C
C...Place The Cursor Below Display Area
C
      CALL TEKSTBM(27,31)
        CALL CUP(27,1)
C
      RETURN
      END
```

```
C*************************************************************************
C*  Routine    C L E A R
C*
C*
C*
C*    Date:                Author:
C*   October 8, 1985   Mark W. Lovell
C*
C*************************************************************************
C*
C*    Description:
C*      This routine clears the scrolling portion of the screen
C*
C*    Calling Sequence:
C*
C*       CLEAR
C*
C*************************************************************************
C
C
      SUBROUTINE CLEAR
C*************************************************************************
C
C   Declarations:
C
C*************************************************************************
C
        INTEGER ERROR
      INTEGER TOPMARGIN/27/
C
C*************************************************************************
C
C   Executable Code
C
C*************************************************************************
C
      CALL CUP(TOPMARGIN,1)
      CALL ED(0,ERROR)
      IF(ERROR)STOP
      RETURN
      END
```

```
C*****************************************************************************
C
C     Block Data program to define common blocks for satellite
C     swath program
C
C*****************************************************************************
C
      BLOCK DATA X
      IMPLICIT COMPLEX(A-Z)
C
      INTEGER SAT
      REAL*8 ORBITDATA(15,7), DTR, PI, STEP, TIME
      REAL*8 A1(200), B1(200), C1(200)
      INTEGER NPOINTS
      REAL LAT,LONG,BEARING,HGHT,DEP,COLAT,P
      INTEGER DIR,FLPATH
C
      COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME,SAT
      COMMON/POINTS/ A1, B1, C1, NPOINTS
      COMMON/TRGDATA/FLPATH,LAT,LONG,BEARING,HGHT,DEP,COLAT,DIR,P
C
      END


      PROGRAM SWATH
      IMPLICIT COMPLEX(A-Z)
C
C*****************************************************************************
C
C   Program    S W A T H
C
C     Date         Author
C  Oct. 9,1985   Suzanne Y. Slinn
C
C*****************************************************************************
C
C   Description:
C      The purpose of this routine is to calculate the satellite
C   swath for an orbit that has been defined by the program SGPSAT
C
C   Calling Sequence:
C      Call Swath
C
C*****************************************************************************
C
C   Declarations:
C
C*****************************************************************************
C
      INTEGER II, I, IOS, IN/5/, OUT/6/, NPOINTS,MAXSAT/15/
      INTEGER JJ,COUNT, SAT,LINE, VERSION_NUM,UNITNO/7/
      INTEGER RECLENGTH/800/,UNITNO1/8/
      REAL*8 GRAZEMIN, GRAZEMAX, TRUEANOM, HEIGHT, SECTORSIZE, PERIOD
      REAL*8 BETAMIN, BETAMAX, PI, MEW, TIME, DTR, A1(200), B1(200)
        REAL*8 C1(200), DLAT(200), CLONG(200), DLTIME
      REAL*8 XP(200), YP(200), ZP(200), INFIN, ARE, LINFIN, ANGLE
      REAL*8 ORBITDATA(15,7), STEP, RTD, EARTHSUR/1.D0/
        REAL*8 SLAT, SLONG, ELAT, ELONG, FILEDATA(15,200,2)
        CHARACTER*12 NAME
        CHARACTER*14 FILENAME
      COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME,SAT
      COMMON/POINTS/A1, B1, C1, NPOINTS
C
C.......Initialization
C
      PI     = 3.1415926536D0
      MEW    = 398601.19D0
      DTR    = PI/180.D0
      RTD    = 180.D0/PI
      INFIN  = 9999.D0
      LINFIN = 99999.D0
C
C...Read In The Orbital Parameters For Each Satellite
```

```
C
      OPEN(UNITNO,FILE = 'SIMDAT.DAT',STATUS = 'OLD', IOSTAT = IOS)
      COUNT = 1
C
      READ(UNITNO,3)DLTIME
3       FORMAT(1X,F9.3)
C
4     IF(COUNT. GT. MAXSAT)GO TO 8
      READ(UNITNO,5,END=8) (ORBITDATA(COUNT,I),I=1,7)
5     FORMAT(1X,7F12.6)
      COUNT = COUNT + 1
      GO TO 4
8     COUNT = COUNT - 1
C
C...Open The Proper SGP Files For Reading In The Satellite Information
C
      DO 15 SAT = 1,COUNT
      VERSION_NUM = 400 + SAT
      NAME = 'SGPSAT.DSP;'
C
      ENCODE(14,10,FILENAME) NAME,VERSION_NUM
10    FORMAT(A11,I3)
        OPEN(UNITNO1+SAT,FILE=FILENAME,STATUS='OLD')
      IF (IOS .NE. 0) THEN
                WRITE(OUT,30)SAT,IOS
 30       FORMAT(' ERROR IN OPENING ORBITAL FILE.',I4,' IOS IS ',I4)
          GO TO 999
      ENDIF
15    CONTINUE
C
C.......Input the grazing angles and sector size of the circle desired
C
      WRITE(OUT,20)
 20   FORMAT('$Input the min, max grazing angles and sector size. )')
      READ(IN,25)GRAZEMIN, GRAZEMAX, SECTORSIZE
 25   FORMAT(3(F6.2,1X))
C
C...Open The Output File
C
      OPEN(UNITNO1,FILE = 'GEOCOR.DAT',STATUS = 'NEW',
     +             RECL = RECLENGTH, IOSTAT = IOS)
        IF (IOS .NE. 0) THEN
         WRITE(OUT,31)IOS
 31       FORMAT(' Error in opening output file. Ios is ',I4)
          GO TO 999
      ENDIF
C
C...Write The Swath Parameters And Number Of Satellites To The GEOCOR.DAT File
C
      WRITE(UNITNO1,35)GRAZEMIN, GRAZEMAX, SECTORSIZE, COUNT
35    FORMAT(3(F6.2,1X),I3)
C
      GRAZEMIN = GRAZEMIN * DTR
      GRAZEMAX = GRAZEMAX * DTR
C
C...Input The Target Parameters
      CALL FLIGHTPATH(DLTIME)
C
C.......Calculate the period of the orbit in order to determine when
C.......one orbit revolution is complete, change to minutes
C
        ARE    = ORBITDATA(1,4)*6378.145D0
      PERIOD = ((2.D0*PI) * SQRT((ARE*ARE*ARE)/MEW))/60.D0
C
      WRITE(OUT,6.)
 82   FORMAT(//,'  Creating The Files...')
C
C.......Repeat for one complete orbit revolution
C
      TIME = 0.D0
      DO WHILE (TIME .LE. PERIOD)
C
C....Read in the time, true anomaly and satellite height For
C....Each Satellite,Calculate One Circle For Each, Store Them
C....In An Array.
```

```
C
      DO 800 SAT = 1,COUNT
         READ(UNITNO1+SAT, 40) TIME, TRUEANOM, HEIGHT
 40      FORMAT(32X,F6.2,18X,F6.2,17X,F7.2)
C
         TRUEANOM = TRUEANOM * DTR
C
C.........Calculate betamin and betamax
C
          CALL CALCBETA(GRAZEMIN, GRAZEMAX, HEIGHT, BETAMIN, BETAMAX)
C
          ANGLE = BETAMIN
        START = 0
          DO 70 JJ = 1, 2
C
C...........Calculate the satellite swath circles
C
          CALL CIRCLE(ANGLE,SECTORSIZE)
C
C...........Transpose from orbit to earth rotating coordinate system
C
           CALL COORDINATETRANSFORMATION(TRUEANOM, DLAT, CLONG)
C
C...........Store The Elements In The Array
C
          DO 60 II = 1, NPOINTS
             FILEDATA(SAT,START+II,1) = DLAT(II)*RTD
             FILEDATA(SAT,START+II,2) = CLONG(II)*RTD
 60       CONTINUE
          FILEDATA(SAT,START+II,1) = INFIN
          FILEDATA(SAT,START+II,2) = INFIN
C
C...Reset Start Point To Load The Second Circle Into The Array
C
          START = START + NPOINTS + 1
C
C...........Save the start and end points of the arc
C
           If (ANGLE .EQ. BETAMIN) THEN
              SLAT  = DLAT(1)
              SLONG = CLONG(1)
              ELAT  = DLAT(NPOINTS)
              ELONG = CLONG(NPOINTS)
           ENDIF
C
C...........Calculate the circle for the other grazing angle
C
           ANGLE = BETAMAX
 70      CONTINUE
C
C.........After both circles have been calculated by swath, join the
C.........end points and then Put In 99999 to indicate end of swath circle
C
          FILEDATA(SAT,START+1,1) = SLAT * RTD
          FILEDATA(SAT,START+1,2) = SLONG * RTD
          FILEDATA(SAT,START+2,1) = Dlat(1) * RTD
          FILEDATA(SAT,START+2,2) = Clong(1) * RTD
          FILEDATA(SAT,START+3,1) = INFIN
          FILEDATA(SAT,START+3,2) = INFIN
          FILEDATA(SAT,START+4,1) = ELAT * RTD
          FILEDATA(SAT,START+4,2) = ELONG * RTD
          FILEDATA(SAT,START+5,1) = DLAT(NPOINTS) * RTD
          FILEDATA(SAT,START+5,2) = CLONG(NPOINTS) * RTD
          FILEDATA(SAT,START+6,1) = INFIN
          FILEDATA(SAT,START+6,2) = INFIN
          FILEDATA(SAT,START+7,1) = LINFIN
          FILEDATA(SAT,START+7,2) = LINFIN
C
C.........Calculate The Circles For The Other Satellites
C
800      CONTINUE
C
C.........Now That Each Satellite Has One Circle, Write The Array To The File
C
```

```
            DO 870 LINE = 1, START + 7
              WRITE(UNITNO1,850) ((FILEDATA(SAT,LINE,II),II = 1,2),
     +                SAT = 1,COUNT)
850           FORMAT((COUNT*2)(1X,F10.3))
870       CONTINUE
C
C.........Calculate Target Position at Particular Time
          CALL TARGETPOS
C
C........Do Another Set Of Circles
C
        END DO
 999    CONTINUE
C
        STOP
        END

        SUBROUTINE CALCBETA(GRAZEMIN,GRAZEMAX,HEIGHT,BETAMIN,BETAMAX)
          IMPLICIT COMPLEX(A-Z)
C
C*************************************************************************
C
C  Subroutine   C A L C B E T A
C
C*************************************************************************
C
        REAL*8 GRAZEMIN, GRAZEMAX, HEIGHT, BETAMIN, BETAMAX, RE
C
        RE = 6378.145D0
C
C........Calculate the half angles of illumination
C
        BETAMIN = ACOS((RE*COS(GRAZEMIN))/(RE+HEIGHT))-GRAZEMIN
        BETAMAX = ACOS((RE*COS(GRAZEMAX))/(RE+HEIGHT))-GRAZEMAX
C
          RETURN
        END


        SUBROUTINE CIRCLE(BETA, SECTORSIZE)
        IMPLICIT COMPLEX(A-Z)
C
C*************************************************************************
C
C   Subroutine   C I R C L E
C
C*************************************************************************
C
        INTEGER I, NPOINTS, SAT
        REAL*8 BETA, SECTORSIZE, A1(200), B1(200), C1(200), RE
        REAL*8 ANGLE, ORBITDATA(15,7), DtR, PI, STEP, TIME
        COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME, SAT
          CoMMON/POINTS/ A1, B1, C1, NPOINTS
C
        RE   = 6378.145D0
        STEP = 10.D0 * DTR
C
C........Calculate the sector of the circle desired
C
        I = 1
        ANGLE = - SECTORSIZE/2.D0*DTR
C
 10         B1(I) = RE * SIN(BETA) * COS(ANGLE)
            C1(I) = RE * SIN(BETA) * SIN(ANGLE)
            A1(I) = RE * COS(BETA)
            ANGLE = ANGLE + STEP
            I = I + 1
          IF (ANGLE .LE. ((SECTORSIZE/2.D0+.01)*DTR)) GO TO 10
C
        NPOINTS = I - 1
        RETURN
        END

        SUBROUTINE COORDINATETRANSFORMATION(TRUEANOM, DLAT, CLONG)
C
```

```
C*********************************************************************
C
C   Subroutine  C O O R D I N A T E    T R A N S F O R M A T I O N
C
C*********************************************************************
C
      INTEGER NPOINTS, I, SAT
      REAL*8  CO, SO, CI, SI, CA, SA, C11, C12, C13, C21, C22, C23
      REAL*8  C31, C32, C33, DTR, PI, TIME , WA, INC, ANA
      REAL*8  TRUEANOM, ALPHA, A1(200), B1(200), C1(200), XP(200)
      REAL*8  YP(200), ZP(200), ORBITDATA(15,7),DLAT(200), CLONG(200)
C
      COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME, SAT
      COMMON/POINTS/ A1, B1, C1, NPOINTS
C
      WA = ORBITDATA(SAT,1)
      INC = ORBITDATA(SAT,3)
      ANA = ORBITDATA(SAT,2)
C
      ALPHA = TRUEANOM + WA
C
C.......Create the components of the transformation matrix
C
      CO  =  COS(ANA)
      SO  =  SIN(ANA)
      CI  =  COS(INC)
      SI  =  SIN(INC)
      CA  =  COS(ALPHA)
      SA  =  SIN(ALPHA)
C
      C11 =  CO*CA - SO*CI*SA
      C12 = -CO*SA - SO*CI*CA
      C13 =  SO*SI
      C21 =  SO*CA + CO*CI*SA
      C22 = -SO*SA + CO*CI*CA
      C23 = -CO*SI
      C31 =  SI*SA
      C32 =  SI*CA
      C33 =  CI
C
C.......Multiply the position vectors by the transformation matrix
C
      DO 1 I = 1, NPOINTS
             XP(I) = C11*A1(I) + C12*B1(I) + C13*C1(I)
             YP(I) = C21*A1(I) + C22*B1(I) + C23*C1(I)
             ZP(I) = C31*A1(I) + C32*B1(I) + C33*C1(I)
1     CONTINUE
C
C.......Change the coordinate system from Earth Centred Inertial to
C.......Earth Rotating
C
      CALL TOEARTHROT(XP, YP, ZP )
C
C.......Change the position vectors to Geodetic Latitude and Longitude
C
      CALL GEODETICLATLONG(DLAT, CLONG)
C
      RETURN
      END

      SUBROUTINE TOEARTHROT(XP, YP, ZP )
      IMPLICIT COMPLEX(A-Z)
C
C*********************************************************************
C
C   Subroutine  T O    E A R T H    R O T
C
C*********************************************************************
C
      REAL*8 XP(200), YP(200), ZP(200), A1(200), B1(200), C1(200)
      REAL*8 ORBITDATA(15,7), DTR, PI, TIME ,CHA, SHA, HOURANG
      REAL*8 EARTHANG, ANGROT/.7292115146D-4/
      INTEGER I, NPOINTS, SAT
C
      COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME,SAT
```

```
      COMMON/POINTS/ A1, B1, C1, NPOINTS
C
C.......Create the components of the transformation matrix
C.......Conpensate for changing hour angle due to earth rotation
C
      HOURANG = ORBITDATA(SAT,7)
      EARTHANG = HOURANG + TIME*ANGROT*60.0D0
      CHA = COS(EARTHANG)
      SHA = SIN(EARTHANG)
C
C.......Transform the position vectors
C
      DO 1 I = 1, NPOINTS
             A1(I) =  CHA*XP(I) + SHA*YP(I)
             B1(I) = -SHA*XP(I) + CHA*YP(I)
             C1(I) =  ZP(I)
 1    CONTINUE
      RETURN
      END

      SUBROUTINE GEODETICLATLONG(DLAT, CLONG)
      IMPLICIT COMPLEX(A-Z)
C
C*********************************************************************
C
C   Subroutine  G E O D E T I C   L A T   L O N G
C
C*********************************************************************
C
      INTEGER I, NPOINTS
      REAL*8  A1(200), B1(200), C1(200)
      REAL*8  CLAT(200), CLONG(200), DLAT(200)
      REAL*8  TEMP, FLAT
C
      COMMON/POINTS/ A1, B1, C1, NPOINTS
C
      FLAT = 1.D0/298.25D0
C
C.......Calculate the geocentric latitude and longitude
C
      DO 1 I = 1, NPOINTS
             TEMP = SQRT((A1(I)*A1(I)) + (B1(I)*B1(I)))
             CLAT(I) = DATAN2(C1(I),TEMP)
             CLONG(I) = DATAN2(B1(I), A1(I))
C            RADIUS(I) = SQRT((A1(I)*A1(I)) + (B1(I)*B1(I)) +
C     1                        (C1(I)*C1(I)))
 1    CONTINUE
C
C.......Calculate the Geodetic Latitude
C
      TEMP = 1-(FLAT*FLAT)
      Do 2 I=1, NPOINTS
             DLAT(I) = DATAN2(TAN(CLAT(I)),TEMP)
 2    CONTINUE
      RETURN
      END

      SUBROUTINE FLIGHTPATH(DLTIME)
      IMPLICIT COMPLEX(A-Z)
C
C*********************************************************************
C
C     Subroutine     F L I G H T   P A T H
C
C        Date                Author
C
C      Feb 1986          Denise Miller
C
C*********************************************************************
C
C   Description:
C        This subroutine gets the beginning values of the parameters to
C        be used in subroutine target_position.
C
C*********************************************************************
```

```
C
C...Declarations
C
      REAL*8 LAT, LONG, BEARING, HGHT, VEL
      REAL*8 COLAT, DEP, ACC, P, KTNM, DLTIME
      INTEGER IN/5/, OUT/6/, FLPATH, DIR
      INTEGER IOS, UNITNO/19/
      INTEGER CLWISE/1/, COCLWISE/2/
      REAL*8 RTD, DTR, PI
      COMMON/TRGTDATA/ FLPATH,LAT,LONG,BEARING,HGHT,DEP,COLAT,DIR,P
C
C...Executable Code
C
      WRITE (OUT,20)
 20   FORMAT(///,' INPUT THE INITIAL PARAMETERS OF THE AIRBORNE TARGET')
      WRITE (OUT,25)
 25   FORMAT(//,'$Latitude and Longitude of Start Position  ) ')
      READ (IN,30)LAT,LONG
 30   FORMAT(2(F7.2,1X))
      WRITE (OUT,35)
 35   FORMAT('$Bearing Angle of Flight Path  ) ')
      READ (IN,40)BEARING
 40   FORMAT(F6.2)
      WRITE (OUT,45)
 45   FORMAT('$Altitude of Target(metres)  ) ')
      READ (IN,60)HGHT
      WRITE (OUT,50)
 50   FORMAT('$Velocity of Target(km/h)  ) ')
      READ (IN,60)VEL
 60   FORMAT(F8.2)
C
C     WRITE (OUT,70)DLTIME
C70   FORMAT('$Delta Time(min)  = ',F9.3)
C
C...Adjust negative values
C
      CALL CHECK_VALUE(LONG)
      CALL CHECK_VALUE(BEARING)
C
C     WRITE(OUT,81)LAT,LONG,BEARING
C81   FORMAT(' ',3(F7.2,x))
C
C...Initialize variables and set flight-path flag
C
      PI    = 3.1415926536D0
      COLAT = 0.D0
      DIR   = 5
      DTR   = PI/180.D0
      RTD   = 180.D0/PI
      ACC   = 0.D0
      KTNM  = 0.5399568D0
C
C.......For flight-path along a meridian
      IF (LAT .EQ. 90.D0)THEN
            LONG = LONG+BEARING
            FLPATH = 2
            CALL CHECK_VALUE(LONG)
C
      ELSE IF (LAT .EQ. -90.D0)THEN
            LONG = LONG+BEARING
            FLPATH = 1
            CALL CHECK_VALUE(LONG)
C
      ELSE IF (BEARING .EQ. 0.D0)THEN
            FLPATH = 1
C
      ELSE IF (BEARING .EQ. 180.D0)THEN
            FLPATH = 2
C
C.......For flight-paths not along a meridian
      ELSE
            COLAT = 90.D0-LAT
            FLPATH = 3
      ENDIF
C
```

```
C...Perform conversion of altitude from metres to km
      HGHT = HGHT/1000.D0
C
C...Perform conversion of velocity from km/hr to nautical miles/min
      VEL = VEL*KTNM/60.D0
C
C...Calculate departure and convert from nautical miles to degrees
      DEP = ((VEL*DLTIME) + (0.5D0*ACC*(DLTIME**2.D0)))/60.D0
C
      IF (BEARING .LT. 180.D0 .AND. BEARING .GT. 0.D0)DIR = CLWISE
      IF (BEARING .GT. 180.D0)DIR = COCLWISE
C
C...Convert to radians
      BEARING = BEARING*DTR
      COLAT   = COLAT*DTR
      P       = DEP*DTR
C
C...Open file to store target position co-ordinates
      OPEN (UNITNO,FILE = 'TRGCOR.DAT', STATUS = 'NEW',IOSTAT = IOS)
      IF (IOS .NE. 0)THEN
              WRITE(OUT,90)IOS
 90           FORMAT(' ERROR IN OPENING TRGCOR.DAT, IOS IS ',I2)
              STOP
      ENDIF
C
      WRITE(UNITNO,95)VEL
 95   FORMAT(X,F8.3)
C
      RETURN
      END

      SUBROUTINE TARGETPOS
      IMPLICIT COMPLEX(A-Z)
C
C*************************************************************************
C
C    Subroutine  T A R G E T   P O S I T I O N
C
C*************************************************************************
C
C  Description:
C       The purpose of this subroutine is to compute the target's
C       positions during the time of the satellite orbit.
C
C*************************************************************************
C
C...Declarations
C
      REAL*8 LAT, LONG, BEARING, HGHT, DEP, COLAT
      INTEGER FLPATH, DIR, I, UNITNO/19/
      INTEGER CLWISE/1/, COCLWISE/2/, SAT
      INTEGER UPMERIDIAN/1/, DOWNMERIDIAN/2/, NONMERIDIAN/3/
      REAL*8 HAVA, A, S, P, DLO, B
      REAL*8 RE, F
      REAL*8 PLANELOC(3), INERTLOC(3)
      REAL*8 DTR, RTD, PI, RLAT, RLONG
      REAL*8 TIME, ORBITDATA(15,7),Q,X
      COMMON/TRGTDATA/ FLPATH,LAT,LONG,BEARING,HGHT,DEP,COLAT,DIR,P
      COMMON/ORBIT/ ORBITDATA,Q,X,TIME,SAT
C
C...Initialize variables
C
      F      = 1.D0/298.25D0
      PI     = 3.141592653D0
      DTR    = PI/180.D0
      RTD    = 180.D0/PI
      RE     = 6378.145D0
C
C.......Store initial target location
      IF (TIME .EQ. 0.D0) THEN
              RLAT = LAT*DTR
              RLONG = LONG*DTR
              GO TO 200
      ENDIF
C
```

```
      IF (FLPATH .EQ. UPMERIDIAN)THEN
              LAT = LAT+DEP
C.............If maximum latitude exceeded,reverse action and begin
C.............subtracting departure
              IF (JIDNNT(LAT) .GT. 90)THEN
                      LAT = LAT-2.D0*DEP
                      FLPATH = DOWNMERIDIAN
                      LONG = LONG+180.D0
                      CALL CHECK_VALUE(LONG)
                      BEARING = BEARING+(180.D0*DTR)
                      CALL CHECK_VALUE(BEARING)
              ENDIF
              RLAT = LAT*DTR
C
      ELSE IF (FLPATH .EQ. DOWNMERIDIAN)THEN
              LAT = LAT-DEP
C.............If minimum latitude exceeded, reverse action and begin
C.............adding departure
              IF (JIDNNT(LAT) .LT. -90)THEN
                      LAT = LAT+2.D0*DEP
                      FLPATH = UPMERIDIAN
                      LONG = LONG+180.D0
                      CALL CHECK_VALUE(LONG)
                      BEARING = BEARING+(180.D0*DTR)
                      CALL CHECK_VALUE(BEARING)
              ENDIF
              RLAT = LAT*DTR
C
      ELSE IF (FLPATH .EQ. NONMERIDIAN)THEN
C.......Calculate new position using oblique spherical triangle method
              HAVA = 0.5D0*(1.D0-COS(BEARING))
              A = ACOS(COS(COLAT-P) - SIN(COLAT)*SIN(P)*2.D0*HAVA)
              S = 0.5D0*(A+COLAT+P)
              DLO = ACOS(1.D0 - 2.D0*SIN(S-A)*SIN(S-COLAT) /
     1                      (SIN(A)*SIN(COLAT)))
              B = ACOS(1.D0 - 2.D0*SIN(S-P)*SIN(S-A) / (SIN(A)*SIN(P)))
C
              COLAT = A
              RLAT = (90.D0*DTR)-A
              IF (DIR .EQ. CLWISE)THEN
                      BEARING = (180.D0*DTR)-B
                      LONG = LONG+(DLO*RTD)
                      CALL CHECK_VALUE(LONG)
              ELSE IF (DIR .EQ. COCLWISE)THEN
                      BEARING = (180.D0*DTR)+B
                      LONG = LONG-(DLO*RTD)
                      CALL CHECK_VALUE(LONG)
              ENDIF
C
      ENDIF
C.....Change to geodetic latitude and longitude
      RLAT = ATAN(TAN(RLAT)*(1.D0-F**2.D0))
      RLONG = LONG*DTR
      LAT = RLAT*RTD
C
  200 CONTINUE
C
c     WRITE (OUT,71)LAT,LONG,BEARING,DIR
c71   FORMAT(2(2X,F7.2),2X,F6.2,2X,I1)
C
C.......Calculate target's x,y,z co-ordinates(in earth-rotating frame)
      PLANELOC(1) = (RE+HGHT)*COS(RLAT)*COS(RLONG)
      PLANELOC(2) = (RE+HGHT)*COS(RLAT)*SIN(RLONG)
      PLANELOC(3) = (RE+HGHT)*SIN(RLAT)
C
C.......Transform co-ordinates to inertial frame
      CALL ROTATE(PLANELOC,INERTLOC)
C
      WRITE(UNITNO,50)FLPATH,TIME,LAT,LONG,BEARING,HGHT,
     1          (PLANELOC(I),I=1,3),(INERTLOC(I),I=1,3)
   50 FORMAT(X,I1,X,F6.2,2(F7.2,X),F8.5,F8.3,6(F12.2,X))
C
C
      RETURN
      END
```

```
C
C*************************************************************************
C
C    Subroutine  C H E C K   V A L U E S
C
C*************************************************************************
C
      SUBROUTINE CHECK_VALUE(VALUE)
C
      REAL VALUE
C
      IF (VALUE .GE. 360.D0) VALUE = VALUE-360.D0
      IF (VALUE .LT. 0.D0) VALUE = VALUE+360.D0
C
      RETURN
      END
C
C
      SUBROUTINE ROTATE(PLANELOC,INERTLOC)
      IMPLICIT COMPLEX(A-Z)
C
C*************************************************************************
C
C    Subroutine  R O T A T E
C
C*************************************************************************
C
      INTEGER OUT/6/, I, SAT
      REAL*8 ANGROT/.0000729D0/, ANG, SA, CA
      REAL*8 HOURANGLE
      REAL*8 PLANELOC(3), INERTLOC(3)
      REAL*8 ORBITDATA(15,7), DTR, PI, TIME
        COMMON/ORBIT/ORBITDATA,DTR,PI,TIME,SAT
C
C...Executable Code
C
C.......Create the components of the transformation matrix
C.......Compensate for changing hour angle due to earth rotation
C
      HOURANGLE = ORBITDATA(1,7)
C
C     WRITE(OUT,7)SAT,ORBITDATA(1,7),HOURANGLE
C 7      FORMAT('  HOURANGLE= ',I1,2(X,F8.5))
C
      ANG = HOURANGLE + TIME*ANGROT*60.0
      CA = COS(ANG)
      SA = SIN(ANG)
C
C.......Transform the position vectors
C
      INERTLOC(1) =  CA*PLANELOC(1) - SA*PLANELOC(2)
      INERTLOC(2) =  SA*PLANELOC(1) + CA*PLANELOC(2)
      INERTLOC(3) =  PLANELOC(3)
C
C     WRITE(OUT,10)TIME,(INERTLOC(I),I=1,3)
C 10  FORMAT(' ',F10.5,' INERTLOC= ',3(F12.2,X))

      RETURN
      END
```

```
$!-----------------------------------------------
$!
$! Procedure  G L I N K
$!
$!   Date:           Author:
$!
$! September 1985   Mark Lovell
$!
$!-----------------------------------------------
$!
$!  Description:
$!      This procedure links all of the modules
$!      needed to run the SATPATH program
$!
$!   Linking Sequence:
$!       @glink
$!
$!-----------------------------------------------
$
$    !GLINK
$
$Link satpath,drawswath,mapper,backmap,graphics,graphics1,-
             drawpath,drawsat,surfaces,getsatdat,drawplane,-
             detect,openfiles,calculateangle,arctangent2,-
             [plotter]plot10/lib
$
$    ![END of GLINK]
```

```
C
C*************************************************************************
C
C   Program    S A T P A T H
C
C     Date          Author
C  September 1985   Suzanne Y. Slinn
C
C*************************************************************************
C
C   Description:
C      The Purpose Of This Program Is To Generate Graphic Output
C       Of Satellite Data Produced By SWATH
C
C*************************************************************************
C
C...Declarations:
C
      PROGRAM MAPTEST
C
      INTEGER CODE,SELECT,STOP,IN,OUT,ERROR
      CHARACTER*1 MAP,SWATHS
C
C....Executable Code
C
      OUT = 6
      IN  = 5
C
C...Set Up Terminal For Graphic Display
C.....Change the terminal to tex mode
C
55    CALL SELECT_CODE(0)
C
C.....Change the size of the input queue
C
      CALL QUEUESIZE(8000)
C
      CALL CLEAR_DIALOG_AREA
C
 10   WRITE (OUT,20)
      WRITE (OUT,25)
 20   FORMAT(/,7X,' SELECT ONE: ',/,10X,'  1)Polar orbit traces ',/,
     1    10X,'  2)Polar ground traces ',/,10X,'  3)Polar target ',
     1    'detection ',/,10X,'  4)Mercator ground traces ',/,10X,
     1    '  5)Mercator target detection ')
 25   FORMAT(//,'$ SELECTION =  ')
      READ(IN,30)SELECT
 30     FORMAT(I1)
C
C...Initialize variables
C
      CODE   = 0
      SWATHS = 'N'
      MAP    = 'P'
C
      IF (SELECT .EQ. 1) THEN
      CODE   = 2
      ELSE IF (SELECT .EQ. 2) THEN
      CODE   = 3
      ELSE IF (SELECT .EQ. 3) THEN
      CODE   = 3
      SWATHS = 'Y'
      ELSE IF (SELECT .EQ. 4) THEN
      CODE   = 1
      MAP    = 'M'
      ELSE IF (SELECT .EQ. 5) THEN
      CODE   = 1
      SWATHS = 'Y'
      MAP    = 'M'
      ELSE
      WRITE(OUT,40)
 40           FORMAT(' Invalid Response')
      GO TO 10
      ENDIF
C
```

```
C...Define Surfaces Which Depends On Whether Or Not Swaths Are Needed
C
      CALL SURFACES(SWATHS)
C
C...Turn Off The Dialog Area
C
      CALL DAVIS(0,ERROR)
      IF (ERROR) STOP
C
C...Draw the correct globe background
C
       CALL BACKMAP(MAP)
C
C...Draw the orbital path on the globe backdrop...
C
      IF (SWATHS .NE. 'Y') THEN
        CALL DRAWPATH(CODE)
C
C...Or Draw swaths if they were requested.
C
      ELSE
        CALL DRAWSWATHS(CODE)
C
      ENDIF
C
C     Turn on the dialog area
C
      CALL DAVIS(1,ERROR)
      IF (ERROR) STOP
C
C     Change queue size back to default
C
      CALL QUEUESIZE(300)
C
C     Change back to ANSII mode
C
      CALL SELECT_CODE(1)
C
      STOP
      END
```

```
C
C*****************************************************************
C
C     Subroutine  D R A W   S W A T H S
C
C      Date          Author
C
C    Oct 1985      Suzanne Y. Slinn
C
C*****************************************************************
C
C Description:
C      This Subroutine Animates The Simulation Data
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE DRAWSWATHS(CODE)
C
      IMPLICIT COMPLEX (A-Z)
      INTEGER I,N1, OLDVIEW, NEWVIEW,COUNT,VERSION_NUM, XY,SAT
      INTEGER ERROR, OFF, ON, OUT, JJ, CODE, IOS, IOSTAT, COUNTER
      INTEGER SOUND, UNITNO1/21/, UNITNO/8/
      INTEGER RESULT, POLAR_GROUND/3/, POLAR_ORBITS/2/, MERCATOR/1/
      REAL XCAR, YCAR, ZCAR, XPLOT, YPLOT, ZPLOT, DTR, RTD, PI
      REAL DATA(500,3,15), INFINTY, EDGELONG/120./, MAX_LATITUDE/85./
      REAL GRAZEMIN, GRAZEMAX, SECTORSIZE
      CHARACTER*14 FILENAME
      LOGICAL*1 EDGE(15),PROB
      COMMON/SWATHS/GRAZEMIN, GRAZEMAX, SECTORSIZE
C
C...Executable Code
C....Initialization
C
      OLDVIEW  = 4
      NEWVIEW  = 3
      OUT      = 6
      OFF      = 0
      ON       = 1
      PI       = 3.1415926
      DTR      = PI/180.
      RTD      = 180./PI
      INFINTY  = 9999.
C
C...Set up the views on the respective surfaces before drawing swaths
C
      CALL VSELECT(3,ERROR)
      IF (ERROR) STOP
      CALL VATTRIB(3,0,1,ERROR)
      IF (ERROR) STOP
      CALL VSELECT(4,ERROR)
      IF (ERROR) STOP
      CALL VATTRIB(4,0,1,ERROR)
      IF (ERROR) STOP
      CALL VSELECT(3,ERROR)
      IF(ERROR) STOP
      CALL SURVIS(3,0,ERROR)
C
C
      CALL INFIN(9998.)
      CALL SETN1
      CALL LINE(0)
      I=1
C
C....Open The File Of Data And Read In The Swath Parameters And The
C....Number Of Satellites That Are On File
C
      OPEN(UNITNO,FILE = 'GEOCOR.DAT',STATUS = 'OLD', IOSTAT = IOS)
      READ(UNITNO,5)GRAZEMIN, GRAZEMAX, SECTORSIZE, COUNT
 5    FORMAT(3(F6.2,1X),I3)
C
C...Open The Proper SSP Files For Reading In The Satellite Location
C....and The File Containing the Target Locations
C
```

```
      CALL OPENFILES(COUNT)
C
C...Open The File Which Will Store Times and Target Positions for
C...Successful Detection
      OPEN(UNITNO1,FILE = 'DETECTPOS.DAT',STATUS = 'UNKNOWN',IOSTAT = IOS)
      IF (IOS .NE. 0)THEN
        WRITE(OUT,130)IOS
 130    FORMAT('  ERROR IN OPENING DETECTPOS.DAT. IOS IS ',I2)
        STOP
      ENDIF
C
C...Read A Line Of Data To Build An Array Of Points To Send To CPLOT.
C...The Storage Of Arrays In FORTRAN Accounts For The Organization
C...Of The Elements
C
 70     READ(UNITNO,90, END=99) ((DATA(I,XY,SAT),XY=1,2),SAT=1,COUNT)
 90     FORMAT((COUNT*2)(1X,F10.3))

C
C.....Determine if one complete circle has been read in
C
      IF (DATA(I,1,1) .EQ. 9999.) THEN
        N1 = I
C
C...Plot Circles For All Satellites
C
      DO 100 COUNTER = 1,COUNT
         CALL NPTS(N1)
         CALL CPLOT(DATA(1,1,COUNTER),DATA(1,2,COUNTER))
 100    CONTINUE
      CALL PLOTEND
        CALL SETN1
        I=1
        GO TO 70
      ENDIF
C
C.....Determine if the two swath circles have been drawn.  If so,get the
C.....current satellite locations, determine if the plane is detected,draw
C.....plane, and satellites(polar view).
C.....Change surfaces, making new picture visible.
C
      IF (DATA(I,1,1) .EQ. 99999.) THEN
       CALL GET_SAT_DAT(COUNT)
       CALL GET_PLANE_DAT
       CALL DETECT_PLANE(COUNT,RESULT)
       IF (CODE .EQ. POLAR_GROUND) CALL DRAWSAT(COUNT)
       CALL DRAWPLANE
         DO 190 SOUND = 1,RESULT
           CALL BEEP
 190     CONTINUE
C
C........Turn off old surface visibility
C
        CALL SURVIS(OLDVIEW,OFF,ERROR)
        IF (ERROR) STOP
C
C....... Run on new surface visibility
C
        CALL SURVIS(NEWVIEW,ON,ERROR)
        IF (ERROR) STOP
C
C........Change the old and new view numbers
C
        OLDVIEW = NEWVIEW
        IF (OLDVIEW .EQ. 3 ) THEN
          NEWVIEW = 4
        ELSE
          NEWVIEW = 3
        ENDIF
C
C........Set 'New' view to current view
C
        CALL VSELECT(NEWVIEW, ERROR)
        IF (ERROR) STOP
C
```

```
C.....Renew the view
C
        CALL RENEWV(NEWVIEW,ERROR)
        IF (ERROR) STOP
C
        GO TO 70
      ENDIF

C
C.....Loop through The Data And Encode The Points For Plotting
C
C
      PROB = .FALSE.
      DO 200 SAT = 1,COUNT
        XCAR = DATA(I,1,SAT)
        YCAR = DATA(I,2,SAT)
        ZCAR = 1.0
        IF (YCAR .GT. 180.0) YCAR = YCAR - 360.0
        XCAR = XCAR * DTR
        YCAR = YCAR * DTR
        CALL PTCOR(4HANG ,XCAR,YCAR,ZCAR,XPLOT,YPLOT,ZPLOT)
C
C....If a Mercator view is desired, indicate to the plotting package to
C....lift the pen at the map boundaries.  The check used for this is as
C....follows:
C....          Determine if the longitude value of the swath is going from
C....          positive to negative .  This indicates that the swath is either
C....          crossing the 0 or 180 degrees of longitude.  In order to
C....          if the 180 degree longitude is being crossed,  it is necessary
C....          to check if the present longitude value is greater than
C....          EDGELONG (120 degrees).
C
          IF (CODE .EQ. MERCATOR)THEN
          DATA(I,1,SAT) = XPLOT
          DATA(I,2,SAT) = YPLOT
          EDGE(SAT) = .FALSE.
          IF (((DATA(I-1,1,SAT)*XPLOT .LT. 0.0) .AND.
     1      (ABS(YCAR) .GT. EDGELONG*DTR)) .OR.
C
C...........Check to see if the latitude value is out of the map boundary
C...........of 85 degrees.
C
     1      (ABS(XCAR) .GE. (MAX_LATITUDE*DTR))) THEN
C
          PROB = .TRUE.
          EDGE(SAT) = .TRUE.
          ENDIF
         ENDIF
C
C.....If polar view being drawn, flag all negative latitude values
C     so that they are not drawn.
C
        IF((CODE .NE. MERCATOR) .AND. (XCAR .LT. 0.0))THEN
         DATA(I,1,SAT) = 9999.
         DATA(I,2,SAT) = 9999.
        ELSE
         DATA(I,1,SAT) = XPLOT
         DATA(I,2,SAT) = YPLOT
        END IF
C
200   CONTINUE
C
C....Now Check For A PROBlem In The Mercator View. If So, Then Lift The
C....Pen But Store The Next Point In The Next Array Element. If Not,
C....Copy the Present Point To Keep The Array The Same Size For All
C....Satellites
C
      IF ( PROB ) THEN
        DO 210 SAT = 1,COUNT
           DATA(I+1,1,SAT)=DATA(I,1,SAT)
           DATA(I+1,2,SAT)=DATA(I,2,SAT)
         IF ( EDGE(SAT) .EQ. .TRUE. )THEN
            DATA(I,1,SAT) = 9999.
            DATA(I,2,SAT) = 9999.
          ENDIF
```

```
210      CONTINUE
            I = I + 1
         ENDIF
C
C.....Return For More Data
C
         I=I+1
         GO TO 70
C
 99      CONTINUE
         CALL PLOTEND
C
         SIZE = 300
C
         RETURN
         END
```

```
C
C*****************************************************************
C
C  Subroutine    B A C K M A P
C
C     Date          Author
C
C   Nov 1, 1985    Mark Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Draws The Proper Backdrop To Orient The
C       Map Drawing Routine
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE BACKMAP( MAP_CODE )
C
      CHARACTER*1 MAP_CODE
C
      REAL TYPE,VLAT,VLONG,LATMIN,LATMAX,LONMIN,LONMAX
      REAL SCALE,LATSTP,LONSTP
      INTEGER DTASTP,ISLLIM,BNDOP,GRID,SHORE,ERR,ERRCR
      COMMON/MAPDATA/TYP,LATMIN,LATMAX,LONMIN,LONMAX
C
C...Executable Code
C...Draw the correct globe background by making the correct segment
C....visible.
C
       CALL VSELECT(1,ERR)
       IF (ERR) STOP
C
       IF ((MAP_CODE .EQ. 'M') .OR. (MAP_CODE .EQ. 'm')) THEN
         CALL SGVIS(2,1,ERR)
         IF(ERR) STOP
       ELSE
         CALL SGVIS(1,1,ERR)
         IF(ERR) STOP
       ENDIF
C
C...Select view two
C
      CALL VSELECT(2,ERR)
      IF (ERR) STOP
C
C...Set view attributes
C
      CALL VATTRIB(2,0,1,ERR)
      IF(ERR) STOP
C
C...Set Up The Map Boundaries In Order To Draw The Orbit Data (For Polar View)
C
      TYP    = 4HORT
      VLAT   = 90.0
      VLON   = 0.
      LATMIN = -90.
      LATMAX = 90.
      LONMIN = -180.
      LONMAX = 180.
      DTASTP = 1000
      ISLLIM = 1000
      BNDOP  = 0
      SCALE  = 2.0
      GRID   = -1
      LATSTP = 10.
      LONSTP = 10.
      SHORE  = 1
C
      IF ((MAP_CODE .EQ. 'M') .OR. (MAP_CODE .EQ. 'm')) THEN
         TYP    = 4HMCR
         VLAT   = 0.
         VLON   = 0.
```

```
            LATMIN = -85.
            LATMAX = 85.
         ENDIF
C
C...Draw The Backdrop, In Colour Index 1
C
         CALL INDEX(1,ERROR)
         IF (ERROR)STOP
         CALL MAP(TYP,VLAT,VLON,LATMIN,LATMAX,LONMIN,LONMAX,
        1 DTASTP,ISLLIM,BNDOP,SCALE,GRID,LATSTP,LONSTP,SHORE)
C
         RETURN
         END
```

```
C
C*****************************************************************
C
C     Subroutine  D R A W   P A T H
C
C      Date          Author
C
C    Nov 4 1985      Mark Lovell
C
C*****************************************************************
C
C Description:
C       This Subroutine Draws Satellite Paths
C
C*****************************************************************
C
C...Declarations
C
        SUBROUTINE DRAWPATH(CODE)

        IMPLICIT COMPLEX (A-Z)
        INTEGER CODE,UNITNO,UNITNO1,ORB,II
        INTEGER IOSTAT,IOS,ERROR,MAXSAT/15/, POLAR_ORBITS/2/,DATANUM/5/
        INTEGER COLOUR_COUNT/1/,FILE_COUNT/1/,ORBIT_COUNT/1/,MAXCOLOUR/7/
        REAL*8 ORBIT_DATA(15,5),STORED_DATA(15,5)
        CHARACTER*12 NAME
        CHARACTER*13 FILENAME
        LOGICAL*1 NEW
C
C...Executable Code
C...Set The Colour
C
       CALL INDEX(COLOUR_COUNT,ERROR)
       IF (ERROR)STOP
C
C...Open The SIMDAT.DAT File And Use It To Identify Which Satellites
C....Are On The Same Orbits. Only One Trace Will Be Drawn For Each
C....Orbit, Not One For Each Satellite.
C
        NEW = .TRUE.
        UNITNO1 = 7
        OPEN(UNITNO1,NAME='SIMDAT.DAT',STATUS='OLD',IOSTAT=IOS)
         IF (IOS .NE. 0) THEN
           WRITE(6,10)IOS
10         FORMAT(' ERROR IN OPENING "SIMDAT.DAT" IOS IS ',I3)
           GO TO 55
       ENDIF
C
C...Loop Through The Data Of The SIMDAT.DAT File, Create An Array
C....Of Parameters Of New Orbits Only. Compare All Records In The
C....File To The Array. If The Orbit Is New, Display It And Add It
C.... To The Array. If It Is Not New, Disregard It.
C
15      IF(FILE_COUNT .GT. MAXSAT) GO TO 55
        READ(UNITNO1,20,END=55)(ORBIT_DATA(FILE_COUNT,II),II = 1,DATANUM)
20      FORMAT(1X,(DATANUM)F12.6)
C
        IF (ORBIT_COUNT .GT. 1)THEN
C
C...Loop Through The Array Of Orbits To See If The Present Orbit Is
C....A NEW Orbit Or Not
C
         ORB = 1
         NEW = .TRUE.
       DO WHILE((ORB .LE. ORBIT_COUNT) .AND. (NEW .EQ. .TRUE.))
         NEW = .FALSE.
C
        DO 30 II = 1,DATANUM
30       IF(ORBIT_DATA(FILE_COUNT,II) .NE. STORED_DATA(ORB,II)) NEW = .TRUE.
C
        ORB = ORB + 1
         END DO
        ENDIF
C
C...If The Orbit Is New, Add It To The Array, Print The Orbit, And Change
```

```
C....The Colour For The Next Orbit
C
       IF (NEW) THEN
      DO 40 II = 1,DATANUM
40    STORED_DATA(ORBIT_COUNT,II) = ORBIT_DATA(FILE_COUNT,II)
C
      ORBIT_COUNT = ORBIT_COUNT + 1
      CALL PRINT_THIS_TRACE(CODE,FILE_COUNT)
C
C...Change The Colour. If The Counter Exceeds The Maximum Colour Index
C....For This Surface Then Reset The Counter To One
C
      COLOUR_COUNT = COLOUR_COUNT + 1
      IF(COLOUR_COUNT .GT. MAXCOLOUR) COLOUR_COUNT = 1
      CALL INDEX(COLOUR_COUNT,ERROR)
        IF (ERROR)STOP
       ENDIF
C
C...Go On To The Next File
C
       FILE_COUNT = FILE_COUNT + 1
       GO TO 15
C
55     CALL PLOTEND
       RETURN
       END

C
C*************************************************************************
C
C    Subroutine  P R I N T _ T H I S _ T R A C E
C
C*************************************************************************
       SUBROUTINE PRINT_THIS_TRACE(CODE,FILE_COUNT)

       IMPLICIT COMPLEX (A-Z)
       INTEGER CODE,FILE_COUNT,UNITNO,VERSION_NUM
       INTEGER IOSTAT,IOS,ERROR,POLAR_ORBITS/2/
       CHARACTER*11 NAME
       CHARACTER*14 FILENAME
C
C...Define NAME And VERSION_NUM For Opening The Proper SGPSAT File
C
       UNITNO = 100
       NAME = 'SGPSAT.DSP;'
       IF (CODE .EQ. POLAR_ORBITS)THEN
          VERSION_NUM = 100 + FILE_COUNT
       ELSE
          VERSION_NUM = 300 + FILE_COUNT
       ENDIF
C
       ENCODE(14,15,FILENAME) NAME,VERSION_NUM
15     FORMAT(A11,I3)
C
       OPEN(UNITNO,NAME=FILENAME,STATUS='OLD',ERR=50,IOSTAT=IOS)
       REWIND UNITNO
       CALL TRACE(CODE,UNITNO)
50     CLOSE(100)
       CALL PLOTEND
       RETURN
       END

C
C*************************************************************************
C
C    Subroutine  T R A C E
C
C*************************************************************************
C
       SUBROUTINE TRACE(CODE ,UNITNO)

       IMPLICIT COMPLEX (A-Z)
       INTEGER I,N1,UNITNO,POLAR_ORBIT/2/,MERCATOR/1/
       INTEGER CODE
       REAL XCAR, YCAR, ZCAR, XPLOT, YPLOT, ZPLOT, DTR, PI, RE
```

```
      REAL X(500), Y(500), Z(500), Height(500)
      REAL INFINTY, MODE, MAG, MAXLATITUDE/85./
      LOGICAL*1 POSITIVE
C
C...Initialization
C
      PI       = 3.1415926
      DTR      = PI/180.
      RE       = 6378.16
      POSITIVE = .TRUE.
      INFINTY  = 9999.
C
      CALL INFIN(9998.)
      CALL SETN1
      CALL LINE(0)
      I=1

 70   IF (CODE .EQ. POLAR_ORBIT) THEN
        READ(UNITNO,80,END=99) XCAR,YCAR,ZCAR
 80     FORMAT(50X,3F12.2)
       MODE = 4HCOR
       XCAR = XCAR/RE
       YCAR = YCAR/RE
       ZCAR = ZCAR/RE
      ELSE
        READ(UNITNO,90,END=99) YCAR,XCAR
 90     FORMAT(56X,F6.2,18X,F6.2)
       MODE = 4HANG
        IF (YCAR .GT. 180.0)YCAR = YCAR - 360.0
        ZCAR = 1.0
        XCAR = XCAR * DTR
        YCAR = YCAR * DTR
      ENDIF
C
      CALL PTCOR(MODE,XCAR,YCAR,ZCAR,XPLOT,YPLOT,ZPLOT)
C
C.....If a Mercator view is desired, check to see if the longitude
C     value is becoming negative going into the western hemisphere
C     only.  If so, lift the pen.
C
       IF (CODE .EQ. MERCATOR)THEN
        IF (((X(I-1)*XPLOT .LT. 0.0) .AND.
     1      XPLOT .LT. 0.0)
     1       .OR. (ABS(XCAR) .GE. (MAXLATITUDE*DTR))) THEN
         X(I) = 9999.
         Y(I) = 9999.
         I = I + 1
       ENDIF
       ENDIF
C
C.....If polar view being drawn, flag all negative latitude values
C     so that they are not drawn.
C
      IF(CODE .NE. MERCATOR) THEN
                MAG = XPLOT**2 + YPLOT**2
       IF( MAG .LT. 1.0 .AND. ZPLOT .LT. 0.0) XPLOT = 9999.
       IF( MAG .LT. 1.0 .AND. ZPLOT .LT. 0.0) YPLOT = 9999.
      END IF
C
      X(I)=XPLOT
      Y(I)=YPLOT
      I=I+1
      GO TO 70
 99   CONTINUE
      N1 = I - 1
C
C...Output Array If Any Points Are Left
C
      IF (N1 .GE. 1) THEN
        CALL NPTS(N1)
        CALL CPLOT(X,Y)
        CALL SETN1
        I=1
      ENDIF
      RETURN
```

```
END
```

```
C
C*****************************************************************
C
C     Subroutine  D R A W S A T
C
C      Date          Author
C
C    Nov 1980      Mark Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Draws The Satellites
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE DRAWSAT(COUNT)
C
      IMPLICIT COMPLEX (A-Z)
      INTEGER I,SAT,COUNT,UNITNO/8/
      REAL XCAR, YCAR, ZCAR
C
C...Executable Code
C
      CALL INFIN(9998.)
      CALL SETN1
      CALL LINE(0)
C
C...Draw A Satellite For Each File
C
      DO 200 SAT = 1,COUNT
        READ(UNITNO+SAT,90) XCAR,YCAR,ZCAR
 90     FORMAT(50X,3F12.2)
      CALL PLOTSAT(XCAR,YCAR,ZCAR)
200   CONTINUE
C
      CALL PLOTEND
      RETURN
      END

C
C*****************************************************************
C
C     Subroutine  P L O T S A T
C
C*****************************************************************
C
      SUBROUTINE PLOTSAT(XCAR,YCAR,ZCAR)

      IMPLICIT COMPLEX (A-Z)
      INTEGER N1
      REAL XCAR, YCAR, ZCAR, XPLOT, YPLOT, ZPLOT
      REAL X(500), Y(500), MODE, MAG, RE
C
C...Executable Code
C
      RE = 6378.14
      N1 = 31
      MODE = 4HCOR
      XCAR = XCAR/RE
      YCAR = YCAR/RE
      ZCAR = ZCAR/RE
C
      CALL PTCOR(MODE,XCAR,YCAR,ZCAR,XPLOT,YPLOT,ZPLOT)
C
      MAG = XPLOT**2 + YPLOT**2
      IF( MAG .LT. 1.0 .AND. ZPLOT .LT. 0.0) GO TO 20
C
      CALL SATELLITE(XCAR,YCAR,X,Y)
      CALL NPTS(N1)
      CALL CPLOT(X,Y)
 20   CALL SETN1
```

```
      RETURN
      END
C
C*************************************************************************
C
C     Subroutine  S A T E L L I T E
C
C*************************************************************************
C
      SUBROUTINE SATELLITE(X,Y,XXS,YYS)

      REAL XX(31),YY(31),XXS(31),YYS(31)
      DATA XX/-6  ,-4  ,-1  ,-2  ,-1  , 1  , 1  , 2  ,
     *          3  , 6  , 4  , 1  , 2  , 1  ,-1  ,-1  ,
     *         -2  ,-3  ,-6  ,-5.5,-2.5,-2  ,-5  ,-4.5,
     *         -1.5,-2  , 5  , 4.5, 1.5, 2.5, 5.3/
      DATA YY/ 4  , 6  , 3  , 2  , 1  , 1  ,-1  ,-2  ,
     *         -1  ,-4  ,-6  ,-3  ,-2  ,-1  ,-1  , 1  ,
     *          2  , 1  , 4  , 4.5, 1.5, 2  , 5  , 5.5,
     *          2.5, 2  ,-5  ,-5.5,-2.5,-1.5,-4.5/
C
C...Executable Code
C
      DO 10 J = 1,31
         XXS(J) = XX(J)/200. + X
         YYS(J) = YY(J)/200. + Y
10    CONTINUE
      RETURN
      END
```

```
C
C*****************************************************************
C
C    Subroutine  S U R F A C E S
C
C     Date          Author
C
C   Nov 1980      Mark Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Sets Up The Graphics Surfaces .
C       4 Surfaces For Moving Display Of Swaths
C       2 Surfaces For Colour Display Of Ground Traces
C
C*****************************************************************
C
C...Declarations
C
        SUBROUTINE SURFACES(CODE)
C
        IMPLICIT COMPLEX (A-Z)
      INTEGER ERROR
        CHARACTER*1 CODE,SWATHS/'Y'/,SWATHS1/'y'/
C
C...Executable Code

C...Define 4 Surfaces For Swath Display
C
      IF (CODE .EQ. SWATHS .OR. CODE .EQ. SWATHS1) THEN
           CALL SDEFN(1,1,1,1,ERROR)
           IF (ERROR) STOP
C
C....Define surface colours
C
           CALL GREEN(1,ERROR)
           IF (ERROR) STOP
           CALL BLUE(2,ERROR)
           IF (ERROR) STOP
           CALL BLUE(3,ERROR)
           IF (ERROR) STOP
           CALL BLUE(4,ERROR)
           IF (ERROR) STOP
C
C...Define 2 Surfaces For Orbits Or Ground Traces
C
      ELSE
           CALL SDEFN(1,3,0,0,ERROR)
           IF (ERROR)STOP
C
C...Set Surface 1 To Green
C
           CALL GREEN(1,ERROR)
           IF (ERROR) STOP
      ENDIF

C
      RETURN
      END
```

```
C
C*****************************************************************
C
C     Subroutine  G E T  S A T  D A T
C
C      Date          Author
C
C    Nov 1985      Mark Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Gets The Satellites' Locations
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE GET_SAT_DAT(COUNT)
C
      IMPLICIT COMPLEX (A-Z)
      INTEGER I,SAT,COUNT,UNITNO/23/
      REAL*8 SATLOC(3,15),TIME
      COMMON/SATDAT/SATLOC
      COMMON/CLOCK/TIME
C
C...Executable Code
C...Read The Current Satellite Location From Each File
C
      DO 200 SAT = 1,COUNT
        READ(UNITNO+SAT,90) TIME,(SATLOC(I,SAT),I=1,3)
 90     FORMAT(38X,4F12.2)
C     WRITE(6,10)(SATLOC(I,SAT),I=1,3)
C10   FORMAT(' ',' SATLOC =   ',3F12.2)
200    CONTINUE
       RETURN
       END

C
C*****************************************************************************
C
C     Subroutine  G E T   P L A N E   D A T
C
C*****************************************************************************
C
      SUBROUTINE GET_PLANE_DAT
      REAL*8 LAT,LONG,HEIGHT,BEARING,T
      REAL*8 PLANELOC(3),INERTLOC(3),TIME
      INTEGER FLPATH,I,UNITNO/19/,OUT/6/
      COMMON/PLANEDAT/PLANELOC
      COMMON/INERTPLANE/INERTLOC
      COMMON/CLOCK/TIME
      COMMON/GEOPLANE/LAT,LONG,HEIGHT,BEARING,FLPATH
C
C...Executable Code
C
      READ(UNITNO,20)FLPATH,T,LAT,LONG,BEARING,HEIGHT,
     1          (PLANELOC(I),I=1,3),(INERTLOC(I),I=1,3)
 20   FORMAT(X,I1,X,F6.2,2(F7.2,X),F8.5,F8.3,6(F12.2,X))
C
C
      IF (T .NE. TIME) THEN
            WRITE(OUT,30)TIME,T
 30         FORMAT('  Time mismatch error found in GETSATDAT.  Time is ',
     1                        f10.5,' Plane time is ',f10.5)
      ENDIF
C
C     WRITE(6,10)(PLANELOC(I),I=1,3),(INERTLOC(I),I=1,3)
C10   FORMAT(' ',' PLANELOC =   ',6F12.2)
C
      RETURN
      END
```

```
C
C*****************************************************************
C
C     Subroutine  D R A W P L A N E
C
C      Date          Author            Revised by          Date
C
C    Nov 1985      Mark Lovell       Denise Miller       Mar 1986
C
C*****************************************************************
C
C Description:
C       This Subroutine Draws The Plane
C
C*****************************************************************
C
C...Declarations
C
        SUBROUTINE DRAWPLANE
C
        IMPLICIT COMPLEX (A-Z)
        INTEGER N1,FLPATH,PLANEPOINTS/5/
        REAL*8 LAT,LONG,HEIGHT,BEARING,PLANELOC(3)
        REAL XCAR, YCAR, ZCAR, XPLOT, YPLOT, ZPLOT
        REAL X(500), Y(500), MODE, MAG, ANGLE, ROT
        REAL PI,DTR,RE
        REAL TYP,LATMIN,LATMAX,LONMIN,LONMAX
        COMMON/GEOPLANE/LAT,LONG,HEIGHT,BEARING,FLPATH
        COMMON/PLANEDAT/PLANELOC
        COMMON/MAPDATA/TYP,LATMIN,LATMAX,LONMIN,LONMAX
C
C...Executable Code
C
        CALL INFIN(9998.)
        CALL SETN1
        CALL LINE(0)
        RE = 6378.145
        ANGLE = 0.0
        ROT  = 0.0
        PI = 3.1415926
        DTR = PI/180.0
C
C...Plot The Plane
C
        N1   = PLANEPOINTS
        MODE = 4HCOR
        XCAR = PLANELOC(1) / RE
        YCAR = PLANELOC(2) / RE
        ZCAR = PLANELOC(3) / RE
C
        CALL PTCOR(MODE,XCAR,YCAR,ZCAR,XPLOT,YPLOT,ZPLOT)
C
C...For orthographic map
C
        IF (TYP .EQ. 4HORTT) THEN
                MAG = XPLOT**2 + YPLOT**2
            IF( MAG .LT. 1.0 .AND. ZPLOT .LT. 0.0) GO TO 20
C.............For flightpath down a meridian
            IF (FLPATH .EQ. 2) THEN
                ROT = (630.0 - LONG)*DTR
                GO TO 12
            ENDIF
C.............For all other flightpaths
            IF (.XCAR .GT. 0.0) .AND. (YCAR .GE. 0.0))THEN
                        ANGLE = 90.0 - LONG
            ELSE
                ANGLE = 450.0 - LONG
            ENDIF
C
C...For mercator map
C
        ELSE IF (TYP .EQ. 4HMCR ) THEN
            IF (ZPLOT .LT. 0.0) GO TO 20
            IF ((LAT .LT. LATMIN) .OR. (LAT .GT. LATMAX)) GO TO 20
```

```
            ANGLE = 180.0
C
        ENDIF
C
        ROT = (ANGLE*DTR) + BEARING
 12     CONTINUE
        CALL PLANE(XPLOT,YPLOT,ROT,X,Y)
C
        CALL NPTS(N1)
        CALL CPLOT(X,Y)
 20     CALL SETN1
        CALL PLOTEND
C
C       WRITE(6,71)LAT,LONG,BEARING,ANGLE
C71     FORMAT(4(F6.2,2X))
C
        RETURN
        END

C
C*************************************************************************
C
C    Subroutine  P L A N E
C
C*************************************************************************
C
        SUBROUTINE PLANE(X,Y,ROT,XXS,YYS)
C
      INTEGER PLANEPOINTS/5/
              REAL ROT,T1,T2
      REAL XX(5),YY(5),XXS(5),YYS(5)
      DATA XX/ 0  , 0  , 5  ,-5 , 0 /
      DATA YY/-7  , 0  , 5  , 5 , 0 /
C
C...Executable Code
C
      T1 = COS(ROT)
      T2 = SIN(ROT)
C
      DO 10 J = 1,PLANEPOINTS
         XXS(J) = ((XX(J)*T1) + (YY(J)*T2))/200. + X
         YYS(J) = ((XX(J)*(-1.0)*T2) + (YY(J)*T1))/200. + Y
10    CONTINUE
      RETURN
      END
```

```
C
C*****************************************************************
C
C     Subroutine  D E T E C T   P L A N E
C
C      Date          Author
C
C    Nov 1985      Mark Lovell
C
C    Modifications:
C    --------------
C
C    March 1986    Suzanne Y. Slinn
C
C*****************************************************************
C
C  Description:
C       This subroutine determines if the plane is detected.
C  Parameters:
C
C    INPUT:  Count  - the number of satellites in the simulation.
C    OUTPUT: Result - the number of satellites that detected the
C                     target at its present location.
C
C  Calling Sequence:
C
C    Call Detect_Plane(Count, Result)
C
C*****************************************************************
C
          SUBROUTINE DETECT_PLANE(COUNT,RESULT)
C
       IMPLICIT COMPLEX (A-Z)
C
C*****************************************************************
C
C  Declarations
C
C*****************************************************************
C
          INTEGER I,FLPATH,SAT,COUNT,RESULT,UNITNO/23/,UNITNO1/21/
       INTEGER IOS,IOSTAT, UNITNO4/60/
          REAL GRAZEMIN, GRAZEMAX, SECTORSIZE
       REAL PGRAZ,ANG
          REAL*8 DIFF(3),INERTLOC(3),SATLOC(3,15), TRUEANOM
          REAL*8 LAT,LONG,HEIGHT,BEARING,TIME
       COMMON/CLOCK/TIME
          COMMON/SATDAT/SATLOC
          COMMON/INERTPLANE/INERTLOC
       COMMON/GEOPLANE/LAT,LONG,HEIGHT,BEARING,FLPATH
          COMMON/SWATHS/GRAZEMIN, GRAZEMAX, SECTORSIZE
C
C*****************************************************************
C
C  Executable Code
C
C*****************************************************************
C
C...Initialize Data
C
       RESULT = 0
         SAT    = 1
C
C...Loop Through All Satellites To See If The Plane Is Detected
C
       DO WHILE( SAT .LE. COUNT )
C
         READ(UNITNO4 + SAT,100)TIME,TRUEANOM,HEIGHT
 100       FORMAT(32X,F6.2,18X,F6.2,17X,F7.2)
C
C
C....Calculate Distance Between The Plane And The Satellite
C
       DIFF(1) = SATLOC(1,SAT) -  INERTLOC(1)
```

```
      DIFF(2) = SATLOC(2,SAT) -  INERTLOC(2)
      DIFF(3) = SATLOC(3,SAT) -  INERTLOC(3)
C
C...Calculate If The Target Is In The Circle Of The Footprint
C
      CALL ANG_BET(INERTLOC,DIFF,ANG)
      PGRAZ = 90 - ANG
      IF (PGRAZ .LE. GRAZEMAX .AND. PGRAZ .GE. GRAZEMIN) THEN
C
C...If Plane Is In The Circle Then See If It Is Within The Sector
C....Move The Satellite Down To The Plane Of The Target
C

         CALL TARGET_ANGLE(SAT,SATLOC(1,SAT),ANG, TRUEANOM)
C
         IF (ANG .LE. SECTORSIZE/2.0) THEN
            RESULT = RESULT + 1
              WRITE(UNITNO1,201)SAT,TIME,LAT,LONG,HEIGHT
 201        FORMAT(' ',I3,2X,F8.2,2(2X,F7.2),2X,F8.3)
            ENDIF
      ENDIF
C
      SAT = SAT + 1
        END DO
C
       RETURN
       END


C*********************************************************************
C
C    Subroutine  T A R G E T    A N G L E
C
C     Date          Author
C
C   Nov 1985      Mark Lovell
C
C   Modifications:
C   -------------
C
C   March 1986   Suzanne Y. Slinn
C*********************************************************************
C
C Description:
C    This routine obtains the angle between the satellite and the
C target.
C
C Parameters:
C
C   INPUT:  SAT         - the number of the present satellite
C           SATL        - a three dimensional array containing the
C                         satellite's position vectors.
C           TRUEANOM    - the true anomaly of the satellite
C
C   OUTPUT: ANG         - the angle between the satellie and the
C                         target in degrees
C
C Calling Sequence:
C
C   Call Target_Angle(Sat, Satl, Angle, Trueanom)
C*********************************************************************
C
       SUBROUTINE TARGET_ANGLE(SAT,SATL,ANGLE, TRUEANOM)
      IMPLICIT COMPLEX (A-Z)
C
C*********************************************************************
C
C Declarations
C
C*********************************************************************
C
       INTEGER I,SAT
```

```
      REAL*8 SATL(3),RIGHTASC(15),NODE(3),INERTLOC(3)
        REAL*8 DISPLACE(3),DIFF(3),ANGLE, TRUEANOM
      REAL*8 ANG,SRATIO,DRATIO,LENGTH,PI,DTR,RTD
      REAL*8 NODE_TO_PLANE,NODE_TO_SAT,SAT_TO_PLANE,TEMP
      REAL*8 ORBSAT(3),ORBTAR(3),ORBNODE(3), NEW_ANGLE
        COMMON/ANG/RIGHTASC
        COMMON/INERTPLANE/INERTLOC
C
C*****************************************************************
C
C Executable code
C
C*****************************************************************
C
C...Initialization
C
        PI  = 3.1415926D0
        DTR = PI/180.D0
        RTD = 1.0D0/DTR
C
C...Find The Vector To The Node
C
      NODE(1) = COS(RIGHTASC(SAT))
      NODE(2) = SIN(RIGHTASC(SAT))
      NODE(3) = 0.D0
C
C...Calculate the angle between the Satellite and Target vectors.
C
        CALL CALCANGLE(SATL,INERTLOC,NODE,TRUEANOM, SAT,
     1    ORBSAT, ORBTAR, ORBNODE, NEW_ANGLE)
C
        ANGLE = NEW_ANGLE * RTD
C
      RETURN
      END

C
C*****************************************************************************
C
C    Subroutine  A N G   B E T
C
C
C Description:
C     This subroutine calculates the angles between the two given
C vectors using the dot product.
C
C*****************************************************************************
C
        SUBROUTINE ANG_BET(A,B,ANG)

        IMPLICIT COMPLEX (A-Z)
      REAL*8 MAG
      INTEGER I
      REAL*8 A(3),B(3)
      REAL*8 PI,RTD,ANG
      REAL*8 DOT,MAG1,MAG2,ARG
C
C...Initialization
C
        PI  = 3.1415926D0
        RTD = 180.D0/PI
C
C...Calculate Dot Product, And The Magnitudes Of The Vectors
C
      DOT = DBLE(A(1)*B(1)+A(2)*B(2)+A(3)*B(3))
      MAG1 = DBLE(MAG(A))
      MAG2 = DBLE(MAG(B))
C
C...Calculate The Angle Between The Vectors
C
      ARG = DOT/(MAG1*MAG2)
      ANG = DACOS(ARG)*RTD
C
      RETURN
      END
```

```
C
C*************************************************************************
C
C    Subroutine  M A G
C
C Description:
C     This routine calculates the magnitude of the given vector.
C
C*************************************************************************
C
        FUNCTION MAG(A)
C
        IMPLICIT COMPLEX (A-Z)
      REAL*8 A(3)
      REAL*8 MAG
C
C...Calculate The Magnitudes Of The Vector
C
      MAG = SQRT(A(1)**2+A(2)**2+A(3)**2)
      RETURN
      END

C
C*************************************************************************
C
C    Subroutine  R O U N D
C
C Description:
C     This routine rounds off the given number to a specified number
C of digits.
C
C*************************************************************************
C
        SUBROUTINE ROUND(NUM)

        IMPLICIT COMPLEX (A-Z)
      REAL*8 NUM
      INTEGER*4 DIGITS
      LOGICAL*1 NEG
C
      NEG = .FALSE.
      IF(NUM .LT. 0.0) THEN
         NEG = .TRUE.
         NUM = DABS(NUM)
      ENDIF
      DIGITS = JIDNNT(NUM*1000.0D0)
      NUM = DBLE(FLOATJ(DIGITS)/1000.0)
      IF (NEG) NUM = NUM * -1.0D0
      RETURN
      END
```

```
C
C*****************************************************************
C
C  Subroutine    O P E N F I L E S
C
C     Date          Author
C
C   November 1985  Mark Lovell
C
C*****************************************************************
C
C Description:
C      This Subroutine Opens The Proper Files For Satellite
C          Position
C
C*****************************************************************
C
C...Declarations
C
      SUBROUTINE OPENFILES(COUNT)
C
      IMPLICIT COMPLEX(A-Z)
      INTEGER COUNT,SAT,VERSION_NUM,VERSION_NUM1,IOS,IOSTAT
      INTEGER UNITNO/9/,UNITNO1723/,UNITNO2739/,UNITNO3/19/
      INTEGER VERSION_NUM2,UNITNO4/60/
      CHARACTER*14 FILENAME
      CHARACTER*12 NAME
      REAL RIGHTASC(15)
      REAL*8 X
      COMMON/ANG/RIGHTASC
C
C...Executable Code
C
      DO 15 SAT = 1,COUNT
C
C...Open The Earth Rotating File
C
        VERSION_NUM  = 200 + SAT
        VERSION_NUM1 = 100 + SAT
C
        ENCODE(14,10,FILENAME) 'SGPSAT.DSP;',VERSION_NUM
10      FORMAT(A11,I3)
          OPEN(UNITNO+SAT,FILE=FILENAME,STATUS='OLD',IOSTAT = IOS)
        IF (IOS .NE. 0) THEN
           WRITE(OUT,13)UNITNO+SAT,IOS
13         FORMAT(' ERROR IN OPENING ORBITAL FILE.',I4,' IOS IS ',I4)
           STOP
        ENDIF
C
C...Open The Inertial File
C
        ENCODE(14,10,FILENAME) 'SGPSAT.DSP;',VERSION_NUM1
          OPEN(UNITNO1+SAT,FILE=FILENAME,STATUS='OLD',IOSTAT = IOS)
        IF (IOS .NE. 0) THEN
           WRITE(OUT,13)UNITNO1+SAT,IOS
           STOP
        ENDIF
C
15    CONTINUE
C
C...Also Open The SIMDAT.DAT File And Store The Right Ascension In The
C....Array RIGHTASC
C
        OPEN(UNITNO2,FILE='SIMDAT.DAT',STATUS='OLD',IOSTAT = IOS)
      IF (IOS .NE. 0) THEN
         WRITE(OUT,30)IOS
30       FORMAT(' ERROR IN OPENING SIMDAT.DAT, IOS IS ',I4)
         STOP
      ENDIF
C
      DO 60 SAT = 1,COUNT
        READ(UNITNO2,50)RIGHTASC(SAT)
50      FORMAT(13X,F12.6)
60    CONTINUE
C
```

```
C...Open The Target Position File
C
      OPEN(UNITNO3,FILE='TRGCOR.DAT',STATUS='OLD',IOSTAT = IOS)
      IF (IOS .NE. 0) THEN
         WRITE(OUT,70)IOS
 70        FORMAT(' ERROR IN OPENING TRGCOR.DAT, IOS IS ',I4)
         STOP
      ENDIF
C
      READ(UNITNO3,75)X
 75          FORMAT(X,F8.4)
C
C...Open the SGP files for reading in the Satellite Information
C
      DO 90 SAT = 1,COUNT
         VERSION_NUM2 = 400 + SAT
         NAME = 'SGPSAT.DSP;'
         ENCODE(14,80,FILENAME)NAME,VERSION_NUM2
 80        FORMAT(A11,I3)
         OPEN(UNITNO4+SAT,FILE=FILENAME,STATUS='OLD')
 90     CONTINUE
C
      RETURN
      END
```

```
C*****************************************************************************
C
C     Block Data program to define common blocks for the angle
C       calculation routine.
C
C*****************************************************************************
C
      BLOCK DATA X
      IMPLICIT COMPLEX(A-Z)
C
      INTEGER SAT
      REAL*8 ORBITDATA(15,7), DTR, PI, STEP, TIME
      REAL*8 A1(200), B1(200), C1(200)
      INTEGER NPOINTS
      REAL LAT,LONG,BEARING,HGHT,DEP,COLAT,P
      INTEGER DIR,FLPATH
C
      COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME
        COMMON/POINTS/ A1, B1, C1, NPOINTS
      COMMON/TRGDATA/FLPATH,LAT,LONG,BEARING,HGHT,DEP,COLAT,DIR,P
C
      END
C
C*****************************************************************************
C
C   Routine C A L C A N G L E
C
C     Date                Author
C  March 19, 1986     Suzanne Y. Slinn
C
C*****************************************************************************
C
C   Description:
C     The purpose of this routine is to calculate the angle between
C   the satellite and target vectors in the orbit plane.
C
C   Calling Sequence:
C      Call Calcangle(Satl, Inertloc, Node, Trueanom, Sat,
C                     Orbsat, Orbtar, Orbnode, Angle)
C
C   Parameters:
C
C     INPUT: Satl     - satellite position vectors in the inertial frame
C            Inertloc - target position vectors in the inertial frame
C            Node     - ascending node position vectors in the inertial frame
C            Trueanom - true anomaly of the satellite in radians
C            Sat      - the number of the current satellite
C     OUTPUT:Orbsat   - satellite position vectors in the orbital frame
C            Orbtar   - target position vectors in the orbital frame
C            Orbnode  - ascending node positon vectors in the orbital frame
C            Angle    - the angle between the target and satellite position
C                       vectors in the orbital frame
C
C*****************************************************************************
C
      SUBROUTINE CALCANGLE(SATL,INERTLOC, NODE,TRUEANOM, SAT,
     1     ORBSAT, ORBTAR, ORBNODE, ANGLE)
      IMPLICIT COMPLEX(A-Z)
C
C*****************************************************************************
C
C   Declarations:
C
C*****************************************************************************
C
      INTEGER II, I, IOS, IN/5/, OUT/6/, NPOINTS,MAXSAT/15/
      INTEGER JJ,COUNT, SAT,LINE, VERSION_NUM,UNITNO/7/
      INTEGER RECLENGTH/800/,UNITNO1/8/
      REAL*8 GRAZEMIN, GRAZEMAX, TRUEANOM, HEIGHT, SECTORSIZE, PERIOD
      REAL*8 BETAMIN, BETAMAX, PI, MEW, TIME, DTR, A1(200), B1(200)
        REAL*8 C1(200), DLAT(200), CLONG(200), DLTIME
      REAL*8 XP(200), YP(200), ZP(200), INFIN, ARE, LINFIN, ANGLE
      REAL*8 ORBITDATA(15,7), STEP, RTD, EARTHSUR/1.D0/
```

```
      REAL*8 SLAT, SLONG, ELAT, ELONG, FILEDATA(15,200,2), NODE(3)
      REAL*8 ORBSAT(3),ORBTAR(3),SATL(3), INERTLOC(3), ORBNODE(3)
      CHARACTER*12 NAME
      CHARACTER*14 FILENAME
      COMMON/ORBIT/ ORBITDATA, DTR, PI, TIME
      COMMON/POINTS/A1, B1, C1, NPOINTS
C
C*****************************************************************************
C
C   Executable Code
C
C*****************************************************************************
C
C...Initialization
C
      PI     = 3.1415926536D0
      MEW    = 398601.19D0
      DTR    = PI/180.D0
      RTD    = 180.D0/Pi
      INFIN  = 9999.D0
      LINFIN = 99999.D0
C
C...Read In The Orbital Parameters For Each Satellite
C
        CLOSE(39)
C
      OPEN(UNITNO,FILE = 'SIMDAT.DAT',STATUS = 'OLD', IOSTAT = IOS)
      COUNT = 1
C
      READ(UNITNO,3)DLTIME
3       FORMAT(1X,F9.3)
C
4     IF(COUNT. GT. MAXSAT)GO TO 8
      READ(UNITNO,5,END=8) (ORBITDATA(COUNT,I),I=1,7)
5     FORMAT(1X,7F12.6)
      COUNT = COUNT + 1
      GO TO 4
8     COUNT = COUNT - 1
      CLOSE(UNITNO)
C
C...Transform the three position vectors from the Inertial frame
C...to the Orbital frame
C
        TRUEANOM = TRUEANOM * DTR
C
        CALL COORDINATETRANSFORMATION(SATL, INERTLOC, NODE,
     1              TRUEANOM,orbtar,orbsat, ORBNODE, SAT)
C
C...Calculate the angle between the target and the satellite in the
C...orbit frame
C
        CALL ARCTANGENT2(ORBTAR(2),ORBTAR(3),ANGLE)
C
 999  CONTINUE
C
      RETURN
      END

C
C*****************************************************************************
C
C   Subroutine  C O O R D I N A T E   T R A N S F O R M A T I O N
C
C      Date:          Author
C   March 1986    Suzanne Y. Slinn
C
C*****************************************************************************
C
C Description:
C     This routine transforms the position vectors in the inertial
C reference frame to the orbital reference frame. The X-axis of the
C orbital frame points in the direction of the satellite motion.
C
```

```
C   Calling Sequence:
C      Call Coordinatetransformation(Satl, Inertloc, Node, Trueanom, Sat,
C                    Orbsat, Orbtar, Orbnode, Angle)
C
C   Parameters:
C
C     INPUT: Satl     - satellite position vectors in the inertial frame
C            Inertloc - target position vectors in the inertial frame
C            Node     - ascending node position vectors in the inertial frame
C            Trueanom - true anomaly of the satellite in radians
C            Sat      - the number of the current satellite
C     OUTPUT:Orbsat   - satellite position vectors in the orbital frame
C            Orbtar   - target position vectors in the orbital frame
C            Orbnode  - ascending node positon vectors in the orbital frame
C            Angle    - the angle between the target and satellite position
C                       vectors in the orbital frame
C*************************************************************************
C
      SUBROUTINE COORDINATETRANSFORMATION(SATL,INERTLOC, NODE,
     1             TRUEANOM,orbinertloc,orbsatl, ORBNODE, SAT)
C
      INTEGER NPOINTS, I, SAT
      REAL*8  CO, SO, CI, SI, CA, SA, C11, C12, C13, C21, C22, C23
      REAL*8  C31, C32, C33, DTR, PI, TIME , WA, INC, ANA, NODE(3)
      REAL*8  TRUEANOM, ALPHA, A1(200), B1(200), C1(200), XP(200)
      REAL*8  YP(200), ZP(200), ORBITDATA(15,7),DLAT(200), CLONG(200)
      REAL*8  ORBINERTLOC(3),ORBSATL(3), SATL(3), INERTLOC(3), ORBNODE(3)
C
        COMMON/ORBIT/ ORBITDATA,DTR,PI,TIME
      COMMON/POINTS/A1,B1,C1,NPOINTS
C
        WA  = ORBITDATA(SAT,1)
      INC = ORBITDATA(SAT,3)
      ANA = ORBITDATA(SAT,2)
      ALPHA = TRUEANOM + WA
C
C...Set up the elements for the transformation matrix
C
      CO  = DCOS(ANA)
      SO  = DSIN(ANA)
      CI  = DCOS(INC)
      SI  = DSIN(INC)
      CA  = DCOS(ALPHA)
      SA  = DSIN(ALPHA)
C
C...Set up the rows of the transformation matrix
C
      C11 = CO*CA - SO*CI*SA
      C12 = SO*CA + CO*CI*SA
      C13 = SI*SA
      C21 =-CO*SA - SO*CI*CA
      C22 =-SO*SA + CO*CI*CA
      C23 = SI*CA
        C31 = SO*SI
      C32 =-CO*SI
      C33 = CI
C
C...Multiply the position vectors by the transformation matrix
C
C...Satellite
C
      ORBSATL(1) = C11*SATL(1) + C12*SATL(2) + C13*SATL(3)
      ORBSATL(2) = C21*SATL(1) + C22*SATL(2) + C23*SATL(3)
      ORBSATL(3) = C31*SATL(1) + C32*SATL(2) + C33*SATL(3)
C
C...Target
C
      ORBINERTLOC(1) = C11*INERTLOC(1)+C12*INERTLOC(2)+C13*INERTLOC(3)
      ORBINERTLOC(2) = C21*INERTLOC(1)+C22*INERTLOC(2)+C23*INERTLOC(3)
      ORBINERTLOC(3) = C31*INERTLOC(1)+C32*INERTLOC(2)+C33*INERTLOC(3)

C
C...Ascending node
C
      ORBNODE(1) = C11*NODE(1) + C12*NODE(2) + C13*NODE(3)
```

```
      CRBNODE(2) = C21*NODE(1) + C22*NODE(2) + C23*NODE(3)
      ORBNODE(3) = C31*NODE(1) + C32*NODE(2) + C33*NODE(3)
C
      RETURN
      END
```

```
C
C*************************************************************************
C
C    Subroutine   A R C T A N G E N T 2
C
C      Date                Author
C
C    March 1986        Suzanne Y. Slinn
C
C*************************************************************************
C
C Description:
C     This routine calculate the angle between the origin of the orbital
C reference frame and the X and Y coordinates of the target position in the
C same plane.
C
C Calling Sequence:
C
C    Call Arctangent2(X, Y, Angle)
C
C Parameters:
C    INPUT:   X     - the X coordinate of the target in the orbital frame
C             Y     - the Y coordinate of the target in the orbital frame
C    OUTPUT:  Angle- the angle between the origin and the specified point.
C
C*************************************************************************
C
      SUBROUTINE ARCTANGENT2(X,Y, ANGLE)
C
      REAL*8 X,Y, ANGLE, ARG, RTD, ARG1
C
        RTD = 180.D0/3.141592654
C
      ANGLE = DATAN2(Y,X)
C
      ANGLE = ABS(ANGLE)
C
        RETURN
      END
```

```
C*****************************************************************************
C
C   Program   E R A S E
C
C   Date              Author
C
C  January 1986    Denise Miller
C*****************************************************************************
C
C   Description:
C        This program will erase the globe, the globe's outline, the
C   satellites and their swaths, and the target after running the program
C   SATPATH.
C
C*****************************************************************************
C
      PROGRAM ERASE
C
      INTEGER VIS/0/,VIEW,SEGMENT,ERROR,OUT/6/
C
C*****************************************************************************
C
C  Executable code
C
C*****************************************************************************
C
      CALL SELECT_CODE(0)
C
C...Erase either view of globe
      VIEW = 1
      CALL VSELECT(VIEW,ERROR)
      IF (ERROR) STOP
      DO 100 SEGMENT = 1,2
          CALL SGVIS(SEGMENT,VIS,ERROR)
          IF (ERROR) STOP
 100    CONTINUE
C
      SEGMENT = 1
      DO 200 VIEW = 2,4
          CALL VSELECT(VIEW,ERROR)
          IF (ERROR) STOP
          CALL SGVIS(SEGMENT,VIS,ERROR)
          IF (ERROR) STOP
          CALL PAGE
          IF (ERROR) STOP
 200  CONTINUE
C
      CALL SELECT_CODE(1)
C
      STOP
      END
```

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATING ACTIVITY Defence Research Establishment Ottawa Department of National Defence Ottawa, Ontario K1A 0Z4	2a. DOCUMENT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. DOCUMENT TITLE
A Simulation Model for Detecting Airborne Targets by Radar

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Note 86-13

5. AUTHOR(S) (Last name, first name, middle initial)
Slinn, Suzanne Y., Lovell, Mark W., Miller, C. Denise

6. DOCUMENT DATE May 1986	7a. TOTAL NO. OF PAGES 208	7b. NO. OF REFS 8
8a. PROJECT OR GRANT NO. 33I	9a. ORIGINATOR'S DOCUMENT NUMBER(S) DREO TN 86-13	
8b. CONTRACT NO	9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)	

10. DISTRIBUTION STATEMENT
Unlimited Distribution

11. SUPPLEMENTARY NOTES	12. SPONSORING ACTIVITY DREO

13. ABSTRACT

In support of the Space Based Radar (SBR) surveillance system technologies, a satellite simulation model has been developed. This simulation models the orbital dynamics of multiple radar satellites, simulates the radar beam in order to study the surveillance coverage capabilities of the satellites and finally, simulates the detection of airborne targets. This report summarizes the major mathematical concepts involved in the model and provides an overview of the computer implementation of this model.

DSIS
77-465